

# Série “Como Fazer”



## Formulários no Excel Utilizando VBA: Multipage e TabStrip

### Parte 2

*por Robert Friedrich Martim*

## Nota sobre direitos autorais

Este eBook é de autoria de Robert F Martim, sendo comercializado através do site [www.juliobattisti.com.br](http://www.juliobattisti.com.br) ou através do site de leilões Mercado Livre: [www.mercadolivre.com.br](http://www.mercadolivre.com.br).

Ao adquirir este eBook você tem o direito de lê-lo na tela do seu computador e de imprimir quantas cópias desejar, desde que sejam para uso pessoal. É vetada a distribuição deste eBook, mediante cópia ou qualquer outro meio de reprodução, para outras pessoas. Se você recebeu este eBook através de e-mail ou via FTP de algum site da Internet, ou através de CD de Revista, saiba que você está com uma cópia pirata, não autorizada. Se for este o seu caso entre em contato com o autor através do e-mail [rm@faircourt.com](mailto:rm@faircourt.com) ou comunique diretamente ao nosso site através do e-mail [webmaster@juliobattisti.com.br](mailto:webmaster@juliobattisti.com.br).

Ao regularizar a sua cópia, você estará remunerando, mediante uma pequena quantia, o trabalho do autor e incentivando que novos trabalhos sejam disponibilizados.

Visite periodicamente o site [www.juliobattisti.com.br](http://www.juliobattisti.com.br) para ficar por dentro das novidades!

## Pré-requisitos

Para completar este curso é necessário um conhecimento médio do ambiente de trabalho do Excel em especial do VBE (Visual Basic Editor). Porém, quanto melhor o seu conhecimento deste ambiente melhor será o aproveitamento deste curso, pois não somente veremos manipulações diretamente no Excel como também acesso a banco de dados.

Este módulo visa a criação de formulários contendo os controles **MultiPage** e **Tabstrip**. Porém, em certas situações outros controles serão utilizados em conjunto para resolver problemas encontrados no nosso dia-a-dia.

Este cursos é a segunda parte do curso sobre formulários no Excel. Se você ainda não possui o primeiro curso ele pode ser adquirido no seguinte endereço <http://www.juliobattisti.com.br/cursos/excelvbaforms/default.asp>.

Os leitores interessados podem adquirir o curso básico de Excel em 120 lições no seguinte endereço: [www.juliobattisti.com.br/excel120/excel120.asp](http://www.juliobattisti.com.br/excel120/excel120.asp) que servirá como trampolim para um melhor aproveitamento e desenvolvimento deste módulo.

## Objetivos deste eBook

Este eBook foi dividido em módulos. Este módulo visa a manipulação dos controles **MultiPage** e **TabStrip** no VBA do Excel.

O trabalho foi desenvolvido a partir da demanda dos usuários do site [www.juliobattisti.com.br](http://www.juliobattisti.com.br). O material procura analisar questões pertinentes ao dia-a-dia de seu trabalho. Vários exemplos são retirados das dúvidas colocadas no fórum do site ou enviadas diretamente para o autor pelos assinantes do site.

A linguagem utilizada é descontraída e com o mínimo de jargão possível. O objetivo é ter um eBook com conteúdo relevante e de fácil compreensão.

Qualquer dúvida referente a este módulo podem ser colocadas diretamente no fórum Excel avançado no endereço: <http://www.juliobattisti.com.br/forum/default.asp>

Comentários e sugestões para melhora do material podem ser enviados diretamente para o autor no endereço [rm@faircourt.com](mailto:rm@faircourt.com)

## ÍNDICE ANALÍTICO

Introdução .....	1
Bem-vindo a série “Como Fazer” .....	1
Antes de continuar .....	1
1. O que são e a diferença entre os controles MultiPage e TabStrip.....	2
2. Devo utilizar um MultiPage ou um TabStrip? .....	5
3. Utilizando o controle Multipage .....	7
3.1. Visão geral das principais propriedades e exemplo de uso do MultiPage .....	7
4. Utilizando o controle TabStrip .....	12
4.1. Visão geral das principais propriedades e exemplo de uso do TabStrip .....	12
5. Criando um sistema de navegação inteligente .....	15
6. Criando um sistema de navegação para avaliar um financiamento .....	19
7. Carregando informações de banco de dados para os controles .....	29
8. Como atualizar o banco de dados central a partir do formulário .....	34
9. Removendo/adicionando informações no banco de dados e criando um sistema de navegação de registros.....	37
10. Criando uma mini prova utilizando um sistema de navegação .....	46
11. Criando um formulário em diferentes idiomas em tempo de execução .....	52
12. Sobre o autor .....	59

## SÉRIES: COMO FAZER

### Formulários no Excel Utilizando VBA: MultiPage e TabStrip

por Robert Friedrich Martim

#### Introdução

##### Bem-vindo a série “Como Fazer”.

Nas séries que serão escritas estaremos olhando aspectos distintos do Excel de acordo com a demanda do fórum Excel Júlio Battisti (<http://www.juliobattisti.com.br>). A intenção principal é fornecer ao internauta uma ferramenta que concentre a atenção na solução de um problema específico.

Esta é a segunda parte da série sobre formulários no Excel. A primeira parte lida com os controles **Listbox** e **Combobox**. Se você não possui a primeira parte ela pode ser adquirida em <http://www.juliobattisti.com.br/cursos/excelvbaforms/default.asp>. Aqui, veremos como manipular os controles **MultiPage** e **TabStrip** de forma eficiente no Excel. Estaremos olhando a maneira como o Excel manipula estes controles e como utilizar este conhecimento para resolver problemas.

##### Antes de continuar

O trabalho foi desenvolvido no Excel 2003. Devido à rápida mudança em termos de tecnologia de software, atenção sempre será dada às versões mais recentes do aplicativo Excel.

Não existe um pré-requisito *per se*; porém, o leitor deve estar ciente de que com o conhecimento básico em dia, o aproveitamento do material será muito melhor. Sem o devido conhecimento de algumas partes básicas que envolvem funções, este módulo se tornará mais laborioso e difícil do que realmente é para aqueles sem o conhecimento básico.

Sugestões serão sempre bem-vindas e esperamos que o leitor participe pro-ativamente no desenvolvimento do material aqui apresentado.

## 1. O que são e a diferença entre os controles MultiPage e TabStrip

Neste primeiro tópico, veremos as diferenças básicas entre um **MultiPage** e **TabStrip**. Embora eles sejam visualmente iguais, estes dois controles são bastante diferentes no que fazem.

Um **MultiPage**, por exemplo, será bastante útil quando precisamos trabalhar com uma grande volume de informações que podem ser classificadas em diversas categorias. Isto geralmente ocorre quando trabalhamos com banco de dados. Vamos supor que estamos lidando com informações cadastrais de funcionários. Utilizando o **MultiPage** podemos agrupar as informações por categorias: uma página poderia conter informações pessoais; outra página relacionaria os interesses do funcionário e assim por diante. A vantagem neste caso é que podemos manter todas as informações sobre o funcionário em um único formulário onde as informações são agrupadas por páginas.

Um **TabStrip** por outro lado contém as mesmas guias (tabulações) que o **MultiPage**, porém os objetos colocados em uma guia são compartilhados com todas as outras guias. Em outras palavras se você colocar uma caixa de texto na primeira guia do **MultiPage**, esta caixa somente estará disponível nesta guia. Já no **TabStrip** a mesma caixa de texto estará presente em todas as guias e exatamente na mesma posição.

Ambos os controles podem ser acessados a partir da caixa de ferramentas do VBE. Caso estes controles não estejam presentes na caixa de ferramentas, você poderá incluí-los da seguinte forma:

1. Clique com o botão direito do mouse sobre a caixa de ferramentas e escolha a opção Controles adicionais;
2. Selecione o controle da lista. Os dois controles são:
  - Microsoft Forms X.x MultiPage
  - Microsoft Forms X.x TabStrip

A figura abaixo mostra a seleção dos dois controles:

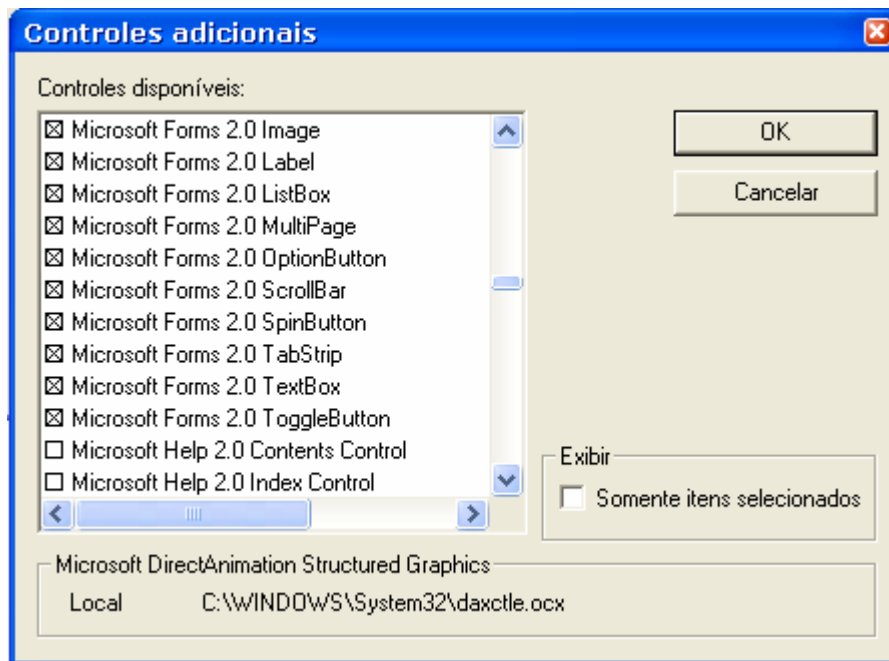


Figura 1-1

Por padrão tanto o **MultiPage** quanto o **TabStrip** inserem 2 tabulações. Tabulações adicionais podem ser acrescentadas através de um clique-direito sobre o objeto em questão. Por exemplo, para acrescentar uma página extra ao **MultiPage**:

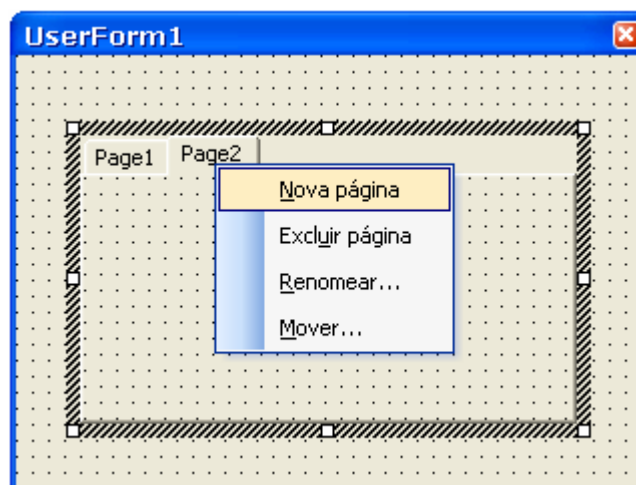


Figura 1-2

Observe que a caixa de propriedades mostra as propriedades para cada página quando ela é selecionada. Se você deseja modificar o **Name** ou **Caption** de uma página qualquer, basta selecionar a página em questão e modificar as propriedades. Se a caixa de propriedades não estiver visível, pressione F4 para ativá-la.

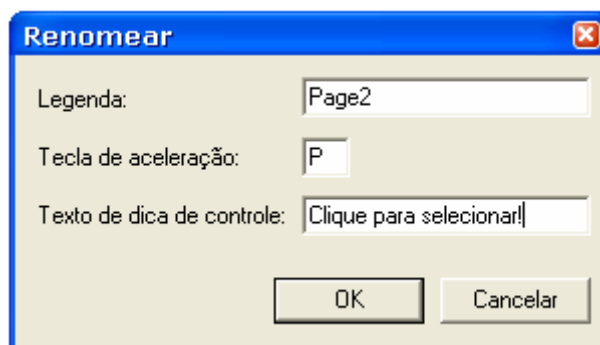
Além da opção Nova página, podemos no mesmo menu de atalho selecionar as seguintes opções:



- Excluir página
- Renomear
- Mover

A primeira opção simplesmente remove a página em questão. A opção **Mover** é utilizada para modificar a posição relativa da página, em outras palavras, a **Page2** pode ser movida para ficar antes da **Page1** e assim sucessivamente.

A opção **Renomear** tem outras opções que precisamos desvendar:



**Figura 1-3**

A propriedade **Legenda** é a propriedade **Caption** na caixa de propriedades e no VBA. A **Tecla de aceleração** serve para criar um atalho de acesso à guia do controle. Neste caso, ao definir a tecla como sendo **P**, a letra **P** na legenda de nossa página será sublinhada indicando que a página pode ser acessada através da combinação **ALT+P**. Este mesmo tipo de tecla de aceleração você encontra nos menus de diversos aplicativos para o Windows.

Em termos de propriedades, ficarei por aqui. O motivo para não entrar nos detalhes das propriedades e que explicações podem ser encontradas no Ajuda (Help) que acompanha o VBA. Em outras palavras, seria repetir aqui o que você encontra gratuitamente no Ajuda.

No caso do uso de propriedades específicas e que necessitem explicações sobre o procedimento, devida atenção será dada ao funcionamento e utilização da propriedade.

## 2. Devo utilizar um MultiPage ou um TabStrip?

Se você alguma vez tentou criar um aplicativo que requer múltiplas páginas, você deve ter encontrado tal dilema, mas como resolvê-lo?

A primeira coisa que precisamos entender é como cada controle funciona. Como observado no tópico anterior, um controle colocado dentro do **TabStrip** estará disponível em todas as tabulações do controle **TabStrip**. Isso significa que o controle não está sendo colocado na guia, mas no container das guias, isto é, no controle **TabStrip**.

Por outro lado, quando você adiciona um controle qualquer em um **MultiPage**, o controle somente estará visível na página onde ele foi adicionado.

Esta diferença indica como devemos escolher entre **MultiPage** e **TabStrip**. Vamos supor que o layout de todas as páginas é igual, mas cada página (guia) mostra valores diferentes. Neste caso, estaremos interessados em um **TabStrip**, pois o layout estará disponível em todas as guias uniformemente.

Porém, se estamos interessados em layouts diferentes em cada guia onde os dados apresentados estão ligados de uma forma ou de outra, estamos interessados em um **MultiPage**.

A figura abaixo mostra a diferença crucial entre os dois controles

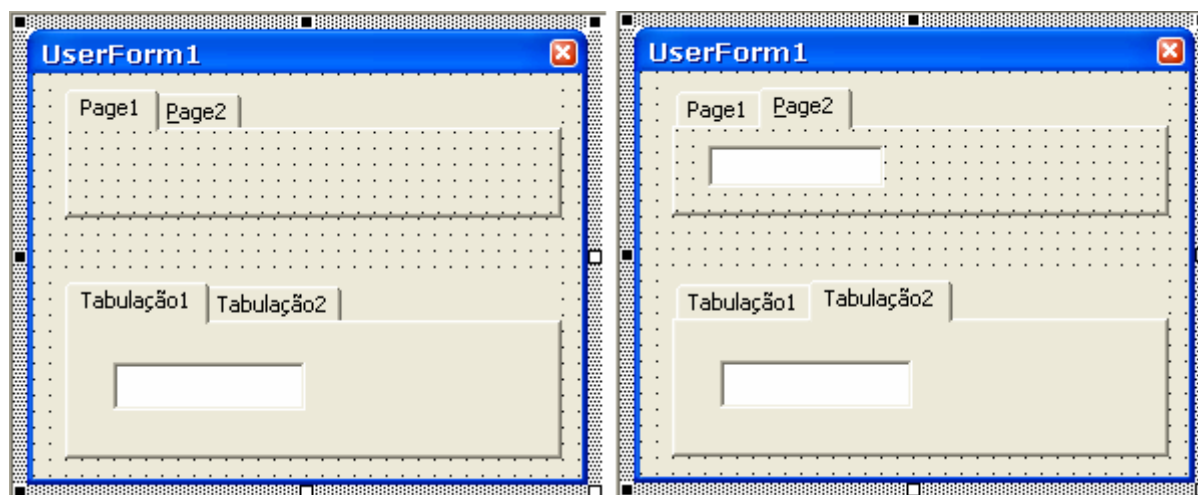


Figura 2-1

Na figura à esquerda temos a seleção na guia 1 (**Page1** e **Tabulação1**)<sup>1</sup> e na figura à direita a seleção da segunda guia. Na segunda página do **MultiPage** tenho uma caixa de texto que

<sup>1</sup> Você deve estar se perguntando o motivo para a não tradução de **Page** e a tradução dos **Tabs**. Esta é uma incógnita que somente os tradutores dos controles poderão sanar, pois não tenho idéia dos motivos para a inconsistência.

somente ficará visível quando a guia for selecionada. Por outro lado, a caixa de texto presente no **TabStrip** é compartilhada nas guias.

Com esta diferença definida a decisão sobre qual escolher começa a tomar forma.

Antes de partirmos para assuntos mais avançados, nos dois próximos tópicos veremos como cada controle separadamente.

### 3. Utilizando o controle Multipage

Para avaliar a utilização destes controles, irei criar alguns exemplos básicos para cada um separadamente.

Acredito que o maior potencial do controle **MultiPage** está em sua versatilidade, isto é, em como podemos criar diversas páginas contendo informações relacionadas sem a necessidade de vários formulários.

Neste primeiro caso, carregaremos dados contidos em nossa planilha para criar nossas páginas. No decorrer do módulo, veremos como carregar estas informações de um banco de dados e manipulá-las.

Já vi formulários com vários controles onde a propriedade **visible** é passada para **true** ou **false** para simular um **MultiPage**. É desnecessário dizer que além de criar código desnecessário controlar tudo isso em um lugar só com tudo amontoado é realmente complicado.

Sem dúvida que nosso conceito do que é bom está sempre em movimento. Hoje, encontramos a solução acima e amanhã inventamos uma forma nova de fazer algo.

Se você alguma vez trabalhou com o Access talvez tenha tido contato com um destes controles. No Excel, ele é realmente pouco conhecido e utilizado.

Nossa primeira para é como mover as tabulações via código.

#### 3.1. Visão geral das principais propriedades e exemplo de uso do MultiPage

Algo que você certamente precisa saber para manipular um **MultiPage** da melhor forma possível diz respeito à navegação pelas tabulações do **MultiPage**.

A chave para isso está na propriedade **Value**. Esta propriedade será revista mais adiante quando construiremos formulários mais avançados.

Cada página de um **MultiPage** recebe uma numeração que vai de zero até o número de páginas inseridas no controle. A figura abaixo mostra um formulário contendo um **MultiPage** com quatro páginas:

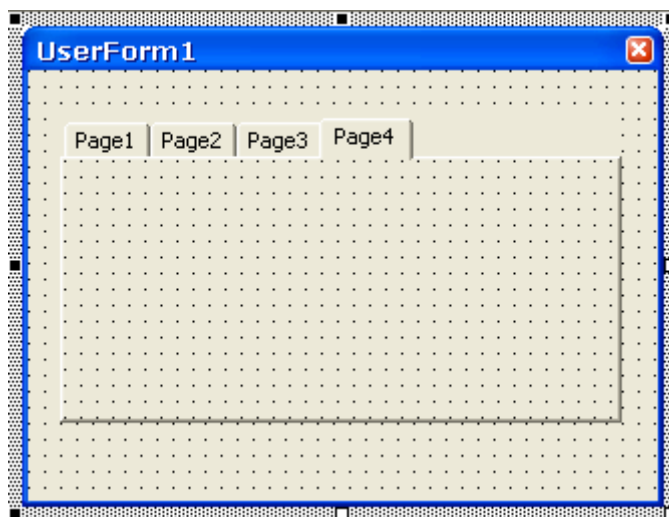


Figura 3-1

Como a quarta página é a selecionada, se este formulário é iniciado esta será a página ativa. Porém, em um sistema de navegação você certamente deseja que a página inicial seja a **Page1**.

A numeração das páginas é iniciada em zero (0) e no caso acima a página quatro recebe o valor 3. Para forçar a inicialização pela primeira página, basta definir o seguinte código em seu formulário:

```
Private Sub UserForm_Activate()  
    MultiPage1.Value = 0    'inicia o MultiPage na 1a. página  
End Sub
```

Ou ainda:

```
Private Sub UserForm_Initialize()  
    MultiPage1.Value = 0    'inicia o MultiPage na 1a. página  
End Sub
```

Uma opção ou outra forçará o início pela **Page1**.

Se você está criando um sistema de navegação no formulário acima com certeza você quer que o formulário seja iniciado pela página um. Além disso, é provável que você queira omitir as tabulações também.

Para omitir as tabulações utilizamos a propriedade **Style**. Esta propriedade pode receber os argumentos **fmTabStyleTabs**, **fmTabStyleButtons** ou **fmTabStyleNone**. A primeira opção é a mostrada na figura acima. A segunda é bem similar ao **botão de alternância (Toggle Button)** e a terceira é a completa remoção das tabulações.

A figura abaixo mostra o controle como botões:

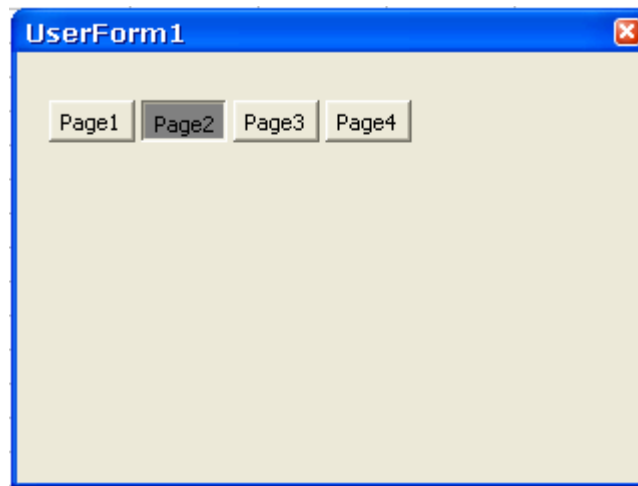


Figura 3-2

Observe que a moldura do **MultiPage** não está mais presente. Se os botões fossem removidos (utilizando a última opção de estilo) o que você obtém é um formulário em “branco”. Ao carregar o formulário, se não houver nenhum outro controle na página do **MultiPage** a impressão que temos é que não há nada no formulário.

A próxima pergunta será: se não há nenhuma tabulação, então, como navegar pelas páginas?

Esta pergunta foi respondida quando definimos a página inicial.

```
Private Sub cmdAvancar_Click()  
    On Error Resume Next  
    MultiPage1.Value = MultiPage1.Value + 1  
End Sub
```

O novo valor do **MultiPage** em caso de avanço será o valor atual do **MultiPage** mais um. A linha **On Error Resume Next** é colocada, pois ao atingirmos o final do controle, no caso acima página quatro valor três, o próximo valor não poderá ser passado para o controle e haverá um erro. Observe a figura:

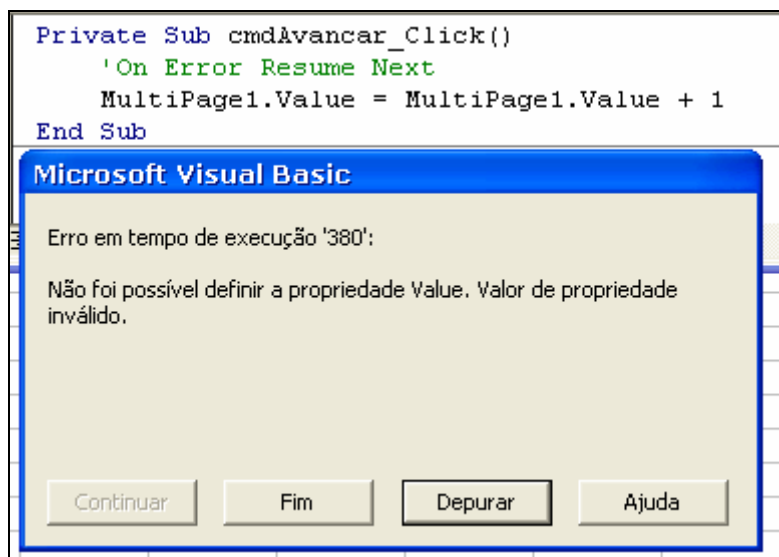


Figura 3-3

Com a linha comentada ocorre um erro durante o avanço. O exemplo acima com o **On Error Resume Next** é apenas um paliativo. Nos tópicos que seguem mostrarei como contar o problema de outra forma.

Por analogia, para voltar uma página subtraímos um do valor atual da página:

```
Private Sub cmdVoltar_Click()
    On Error Resume Next
    MultiPage1.Value = MultiPage1.Value - 1
End Sub
```

Uma propriedade que pode ser bastante útil é a **MultiRow**. Se você já utilizou o caixa de opções do Excel você certamente já viu isso:

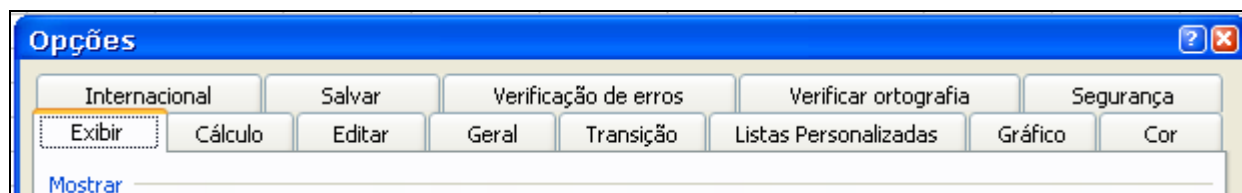


Figura 3-4

A figura acima mostra a caixa de opções do Excel com seu **MultiPage** com tabulações como se fossem as abas de etiquetas de uma pasta de arquivo. Este efeito é conseguido quando modificamos a propriedade **MultiRow** para **True** (verdadeiro):

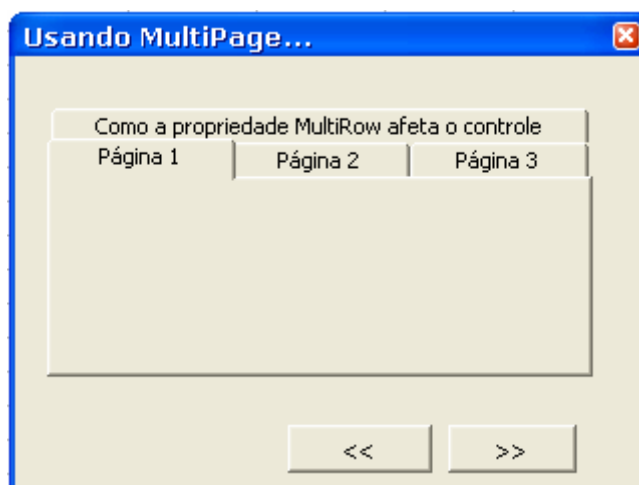


Figura 3-5

A figura acima mostra o que ocorre ao modificarmos esta propriedade. Como pode ser observado, o resultado final é dependente do tamanho do **MultiPage** e de cada tabulação dentro do **MultiPage**.

Se o objetivo de seu **MultiPage** é similar ao opções do Excel é interessante utilizar esta propriedade para que você possa fazer melhor proveito do espaço em seu formulário.



## 4. Utilizando o controle TabStrip

O controle **TabStrip**, como observado inicialmente, é bem similar ao **MultiPage**. A diferença é que um controle colocado em uma página do **TabStrip** estará presente em todas as outras páginas.

Esta característica pode representar uma grande vantagem. Supondo que os campos são exatamente os mesmos para duas páginas onde somente os dados mudam, então, devemos usar um **TabStrip** ao invés de um **MultiPage**, pois o design é constante em todas as páginas.

Um exemplo interessante é o exemplo padrão dado pela própria Microsoft. O exemplo consiste em um formulário com um **TabStrip** com três tabulações com o nome de três cores. No **TabStrip** há uma moldura de imagem e a moldura muda de cor conforme selecionamos uma tabulação diferente.

Este exemplo, de domínio público, acompanha este módulo, mas estarei mostrando o meu próprio exemplo.

### 4.1. Visão geral das principais propriedades e exemplo de uso do TabStrip

Tanto o **MultiPage** quanto o **TabStrip** possuem várias propriedades em comum e a diferença será notada quando você seleciona uma página no **TabStrip** e outra no **MultiPage**. Esta diferença ocorre pela própria natureza de um **TabStrip**. Se fosse construir um formulário contendo vários botões no topo simulando um **TabStrip**, você verá que o resultado é basicamente o mesmo, pois o **TabStrip** não funciona como um contenedor do objeto como o **MultiPage**.

Para iniciar o seu formulário com o índice do **TabStrip** na primeira página, você deve utilizar o mesmo esquema que o **MultiPage**:

```
Private Sub UserForm_Activate()  
    TabStrip1.Value = 0    'inicia o TabStrip na 1a. página  
End Sub
```

Ou ainda:

```
Private Sub UserForm_Initialize()  
    TabStrip1.Value = 0    'inicia o MultiPage na 1a. página  
End Sub
```

Uma opção ou outra forçará o início pela **Tab1**.

O exemplo a seguir mostra como utilizar o **TabStrip** em uma situação onde você tem várias planilhas que representam estantes onde produtos são guardados. Cada estante possui suas prateleiras onde os produtos são guardados.

O nosso formulário possui um **TabStrip** com três tabulações. Cada tabulação representa uma planilha. Dentro do **TabStrip** temos duas caixas de texto onde cada uma recebe a prateleira onde se encontra o produto e o nome do produto. Dentro do mesmo **TabStrip** há dois botões para navegação dentro da estante, isto é, navegação pelos produtos.

O layout geral do formulário será:

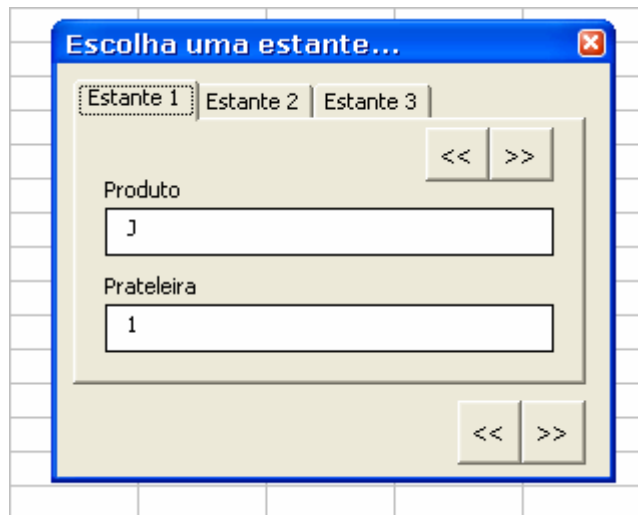


Figura 4-1

Com o layout do formulário pronto, podemos iniciar o nosso código:

```
Private Sub UserForm_Activate()  
    tbsEstantes.Value = 0  
    txtPrateleira = Sheets("Estante 1").Range("A2")  
    txtProduto = Sheets("Estante 1").Range("B2")  
End Sub  
  
Private Sub tbsEstantes_Change()  
    On Error Resume Next  
    Call atualizar(tbsEstantes.SelectedItem.Caption, 2)  
End Sub  
  
Private Sub cmdAvancar_Click()  
    On Error Resume Next  
    ' Move para a próxima estante  
    tbsEstantes.Value = tbsEstantes.Value + 1  
    ' Pega o primeiro item da estante e mostra-os na caixa de texto  
    Call atualizar(tbsEstantes.SelectedItem.Caption, 2)  
End Sub
```

```
Private Sub cmdAvancarPrateleira_Click()  
    ' Procura pelo valor atual na lista  
    ' A lista vai de B1 a B100. Você pode usar um nome  
    ' dinâmico se preferir  
    lin = WorksheetFunction.Match(txtProduto, _  
        Sheets(tbsEstantes.SelectedItem.Caption).Range("B1:B100"), 0) + 1  
  
    ' Se a próxima linha não for vazia, então  
    If Not Range("B" & lin) = "" Then  
        ' Atualizar os dados  
        Call atualizar(tbsEstantes.SelectedItem.Caption, lin)  
    Else:  
        ' Se a próxima linha for vazia, os registros terminaram  
        ' Avisar o usuário e ...  
        MsgBox "Este foi o último registro encontrado. " _  
            & "Voltando ao primeiro registro.", vbInformation + vbOKOnly  
        ' ... voltar para o primeiro item.  
        Call atualizar(tbsEstantes.SelectedItem.Caption, 2)  
    End If  
End Sub
```

```
Private Sub cmdVoltar_Click()  
    On Error Resume Next  
    tbsEstantes.Value = tbsEstantes.Value - 1  
    Call atualizar(tbsEstantes.SelectedItem.Caption, 2)  
End Sub
```

```
Private Sub atualizar(plan As String, ByVal lin As Long)  
    ' Recebe os parâmetros dos cliques e atualiza o registro  
    txtProduto = Sheets(plan).Range("B" & lin)  
    txtPrateleira = Sheets(plan).Range("A" & lin)  
End Sub
```

```
Private Sub cmdVoltarPrateleira_Click()  
    lin = WorksheetFunction.Match(txtProduto, _  
        Sheets(tbsEstantes.SelectedItem.Caption).Range("B1:B100"), 0) - 1  
    If Not lin = 1 Then  
        Call atualizar(tbsEstantes.SelectedItem.Caption, lin)  
    Else:  
        MsgBox "Este é o primeiro registro da lista.", _  
            vbInformation + vbOKOnly  
        Call atualizar(tbsEstantes.SelectedItem.Caption, 2)  
    End If  
End Sub
```

## 5. Criando um sistema de navegação inteligente

Se você trabalha com Excel em algum ponto você deve ter passado por uma situação que requer a criação de um Assistente. Assistentes são extremamente úteis, pois eles nos permitem guiar o usuário para que a resposta final seja realmente o que desejamos. Para o usuário, o Assistente é benéfico por tornar a construção do problema mais fácil.

Nesta parte estarei demonstrando como criar um sistema de navegação inteligente. Primeiramente, estarei demonstrando os passos principais e no final de tópico apresentarei uma solução que pode ser aplicada ao problema de avaliação de crédito.

Para iniciar crie um novo formulário e acrescente um **MultiPage** (acrescentando controles adicionais em cada página) e dois botões<sup>2</sup>, conforme a figura abaixo:

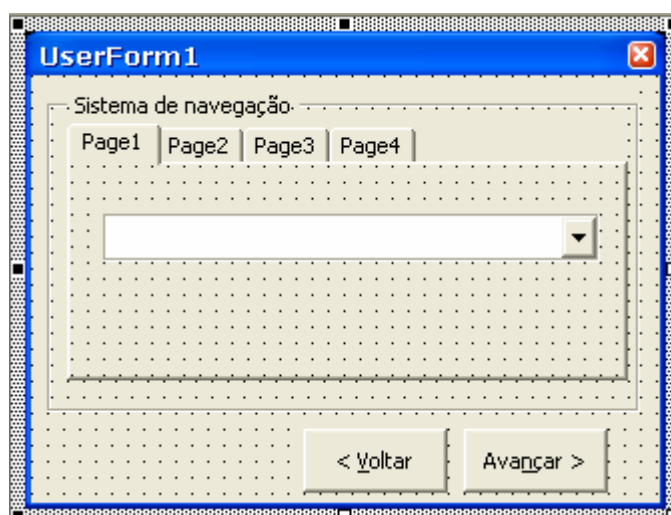


Figura 5-1

Com as páginas criadas, podemos passar a propriedade **Style** para **fmTabStyleNone** para remover as tabulações das páginas. Você pode modificar esta propriedade a qualquer momento, mas o ideal é remover as tabulações ao terminar todo o código e design para facilitar acesso às páginas em caso de necessidade durante o desenvolvimento.

Iniciarei pelo código do formulário, isto é, quando o formulário for ativado algo acontecerá:

```
Private Sub UserForm_Activate()  
    MultiPage1.Value = 0 'inicia o MultiPage na 1a. página  
    chkPaginação        'checa as condições  
End Sub
```

<sup>2</sup> Se desejar, acrescente um valor para a propriedade **Accelerator** dos botões. Neste caso, utilizo a letra “V” e “n” para acelerar o acesso aos dois controles via teclado.

A primeira coisa a ser feita é assegurar que iniciamos na primeira página. Em seguida, rodamos a rotina **chkPaginação**. O que esta rotina faz é bastante simples. Imagine a situação onde estamos no limite do **MultiPage**? Se você clicar no botão **Voltar** haverá um erro, pois não há mais nada para a esquerda. Por outro lado, se você deseja avançar e não há mais páginas para avançarmos; então, haverá um erro também.

A solução é encontrar o fim da paginação é desativar o botão em questão, isto é, se estamos no limite esquerdo desativamos o botão **Voltar** e se estamos no limite direito desativamos o botão **Avançar**. Abaixo apresento duas formas distintas de se fazer a mesma coisa:

```

Sub chkPaginação()
    índice = MultiPage1.Value
    fim = MultiPage1.Pages.Count - 1

    Select Case índice
        Case 0
            cmdVoltar.Enabled = False
            cmdAvançar.Enabled = True

        Case fim
            cmdVoltar.Enabled = True
            cmdAvançar.Enabled = False

        Case Else
            cmdVoltar.Enabled = True
            cmdAvançar.Enabled = True
    End Select

    ' .....
    ' VOCE PODE UTILIZAR AS CONDICÕES ABAIXO PARA SIMULAR
    ' AS CONDICÕES DO BLOCO SELECT CASE-END SELECT
    ' APRESENTADAS ACIMA
    ' .....
    'If índice = 0 Then
    '    cmdVoltar.Enabled = False
    '    cmdAvançar.Enabled = True
    '    Exit Sub
    'End If

    'If índice = fim Then
    '    cmdVoltar.Enabled = True
    '    cmdAvançar.Enabled = False
    'Else:
    '    cmdVoltar.Enabled = True
    '    cmdAvançar.Enabled = True
    'End If

End Sub

```

Aqui, avalio o caso referente ao índice atual do **MultiPage**. Cada página é representada por um número e este número é passado para variável **índice**. Uma outra variável importante é a referente ao final da paginação. A variável **fim** é encontrada contando-se o número total de páginas e subtraindo um deste total. Isto é necessário porque a primeira página tem índice 0 (zero); portanto a contagem resultará no número correto de páginas, mas com um índice a mais.

Resolvido este problema, passamos para os botões de navegação.

```
Private Sub cmdAvançar_Click()  
    'Denife o novo valor do MultiPage como sendo  
    'o valor atual MAIS 1  
    MultiPage1.Value = MultiPage1.Value + 1  
  
    'checa a paginação para definir o estado dos  
    'botões de navegação  
    chkPaginação  
End Sub  
  
Private Sub cmdVoltar_Click()  
    'Denife o novo valor do MultiPage como sendo  
    'o valor atual MENOS 1  
    MultiPage1.Value = MultiPage1.Value - 1  
  
    'checa a paginação para definir o estado dos  
    'botões de navegação  
    chkPaginação  
End Sub
```

Com estes dois últimos códigos prontos, terminamos o nosso sistema de navegação. A figura abaixo mostra a situação onde nos encontramos nos extremos de nosso formulário:

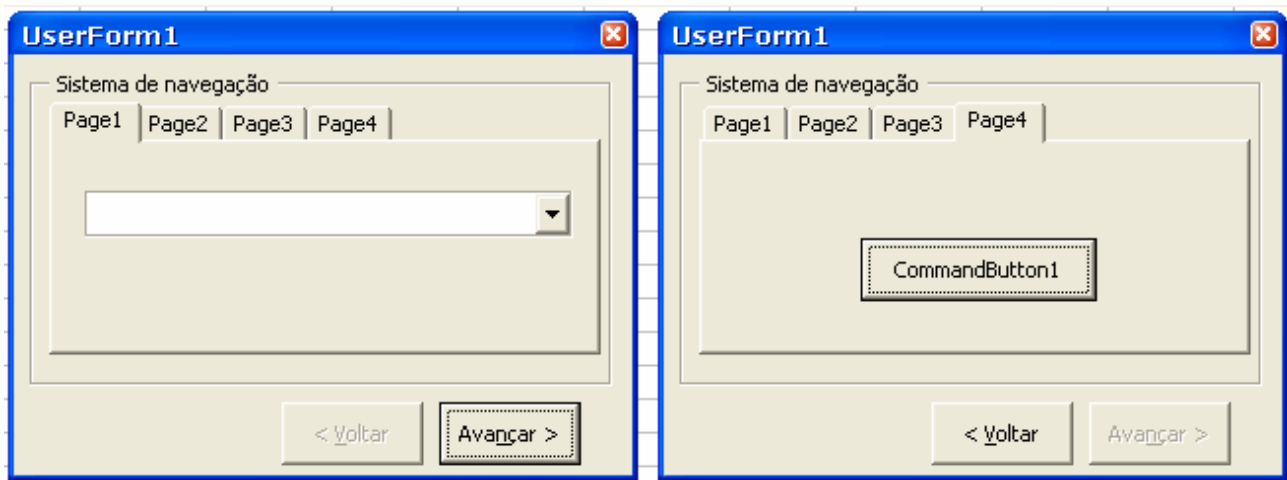


Figura 5-2

Ao iniciar o botão **Voltar** é desativado por não haver mais páginas à esquerda. No outro extremo, o botão avançar é desativado por não haver mais páginas para avançarmos. Ao removermos as tabulações o efeito criado pelo sistema é ainda melhor:

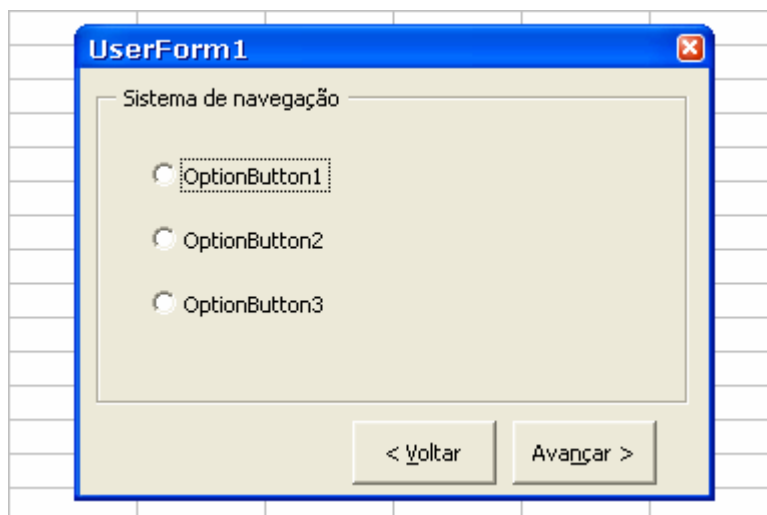


Figura 5-3

Na figura acima, naveguei até a terceira página. Como o **MultiPage** encontra-se dentro de uma **Moldura** a transição entre páginas acontece sutilmente e todos os controles parecem estar dentro da **Moldura**.

A nossa próxima tarefa é criar um Assistente para avaliação de crédito. No problema que definirei, nós estamos interessados em saber informações pessoais do indivíduo, informações de trabalho, informações sobre ganhos e produto a ser comprado e financiado.

Tudo isso ocorre diretamente no **MultiPage** onde utilizamos o sistema de navegação apresentado anteriormente. No último passo é retornada uma tabela contendo os valores que foram calculados do financiamento.

## 6. Criando um sistema de navegação para avaliar um financiamento

Depois de mostrar como um sistema de navegação pode ser criado, chegou a hora de criar um pequeno sistema para avaliação de crédito utilizando o que foi aprendido.

Aqui, estou assumindo que você possui certo conhecimento de matemática financeira. Porém, mesmo não tendo o desenvolvimento do problema é bastante simples e fácil de acompanhar.

Para iniciar crie três planilhas. Nomeie uma de Input, onde inseriremos os dados de entrada do formulário e a outra de Resultados, onde os resultados dos cálculos serão colocados.

A primeira terá os seguintes dados (veja pasta de trabalho que acompanha este tópico):

	A	B	C	D	E	F	G	H
1	UF						PRODUTO	PREÇO
2	AC						HOME TRANSPORT CALOI CL HT04 2004	R\$ 709.90
3	AL						MICROONDAS CONSUL CMS 180 18L	R\$ 399.00
4	AM						MICRO SYSTEM SEMPTOSHIBA MS 6531	R\$ 659.00
5	BA						LAVADORA ROUPAS BRASTEMP TIRA MANCHAS BWT08	R\$ 1,459.00
6	CE						TELEVISOR LG MP-42PZ45V	R\$ 19,999.00
7	DF							

Figura 6-1

E a planilha de resultados:

	A	B	C	D	E
1	Mês	Pgto	Juros	Principal	Balanco devido
2	0				R\$ 19,999.00
3	1	R\$ 526.65	R\$ 199.99	R\$ 326.66	R\$ 19,672.34
4	2	R\$ 526.65	R\$ 196.72	R\$ 329.93	R\$ 19,342.41
5	3	R\$ 526.65	R\$ 193.42	R\$ 333.23	R\$ 19,009.18
6	4	R\$ 526.65	R\$ 190.09	R\$ 336.56	R\$ 18,672.62
7	5	R\$ 526.65	R\$ 186.73	R\$ 339.92	R\$ 18,332.70
8	6	R\$ 526.65	R\$ 183.33	R\$ 343.32	R\$ 17,989.38
9	7	R\$ 526.65	R\$ 179.89	R\$ 346.76	R\$ 17,642.62
10	8	R\$ 526.65	R\$ 176.43	R\$ 350.22	R\$ 17,292.40
11	9	R\$ 526.65	R\$ 172.92	R\$ 353.73	R\$ 16,938.67
12	10	R\$ 526.65	R\$ 169.39	R\$ 357.26	R\$ 16,581.41
13	11	R\$ 526.65	R\$ 165.81	R\$ 360.84	R\$ 16,220.57
14	12	R\$ 526.65	R\$ 162.21	R\$ 364.44	R\$ 15,856.13

Figura 6-2

Os resultados são inseridos pelo cálculo e a figura acima é apenas ilustrativa.

Crie um formulário e adicione um **MultiPage** com cinco páginas. Cada página deverá conter os controles conforme as figuras:





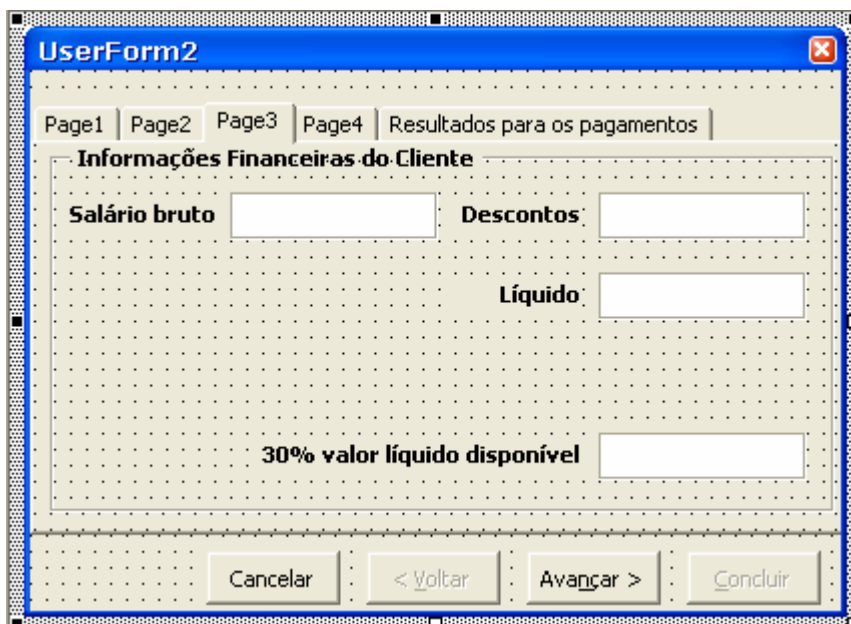
The image shows a VBA UserForm titled "UserForm2" with a tabbed interface. The tabs are "Page1", "Page2", "Page3", "Page4", and "Resultados para os pagamentos". The "Page1" tab is selected. The form contains a section titled "Informações pessoais do cliente" with the following fields: "Nome" (text box), "Endereço" (text box), "CEP" (text box), "Cidade" (text box), "UF" (dropdown menu), "Fone" (text box), and "Celular" (text box). At the bottom, there are four buttons: "Cancelar", "< Voltar", "Avançar >", and "Concluir".

Figura 6-3 – Página 1



The image shows the same VBA UserForm "UserForm2" but with the "Page2" tab selected. The section title is "Informações sobre o empregador do cliente". The fields are: "Nome" (text box), "Endereço" (text box), "CEP" (text box), "Cidade" (text box), "UF" (dropdown menu), "Fone" (text box), and "Fax" (text box). The bottom buttons remain the same: "Cancelar", "< Voltar", "Avançar >", and "Concluir".

Figura 6-4 – Página 2



UserForm2

Page1 Page2 Page3 Page4 Resultados para os pagamentos

**Informações Financeiras do Cliente**

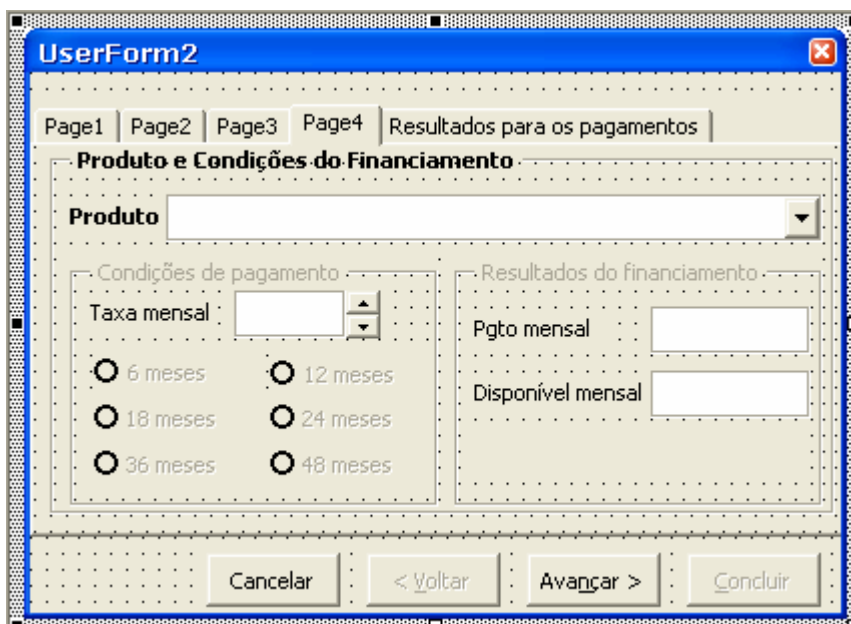
Salário bruto  Descontos

Líquido

30% valor líquido disponível

Cancelar < Voltar Avançar > Concluir

Figura 6-5 – Página 3



UserForm2

Page1 Page2 Page3 Page4 Resultados para os pagamentos

**Produto e Condições do Financiamento**

Produto

Condições de pagamento

Taxa mensal

☐ 6 meses ☐ 12 meses

☐ 18 meses ☐ 24 meses

☐ 36 meses ☐ 48 meses

Resultados do financiamento

Pgto mensal

Disponível mensal

Cancelar < Voltar Avançar > Concluir

Figura 6-6 – Página 4

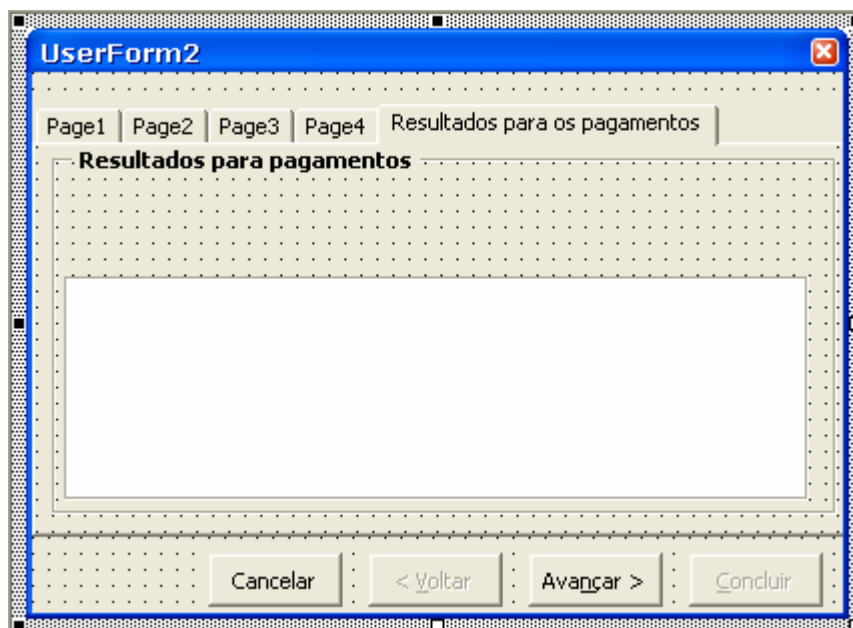


Figura 6-7 – Página 5

Para a página 5, o controle a ser acrescentado é uma **ListBox** (**Caixa de Listagem**) que será utilizada para colocar o resultado de nossos cálculos.

Os controles de navegação devem estar fora do **Multipage** e para dar um ar melhor ao nosso projeto, utilize uma **Moldura (Frame)** para agrupar os controles de cada página.

Após posicionar todos os seus controles e estiver satisfeito com o layout geral de seu formulário e páginas do **MultiPage**, modifique a propriedade **Style** do **MultiPage** para **fmTabStyleNone** para esconder as tabulações, obtendo um resultado final como mostra a figura abaixo:

Figura 6-8

O layout final fica, obviamente, a critério do leitor. Ele não precisa ser exatamente como apresentado. Dê o seu próprio toque pessoal ao formulário. O melhor é “brincar” com diferentes layouts para ver qual se encaixa melhor na mensagem e ajuda que você deseja passar e dar ao usuário de seu formulário.

Dito isso, estamos prontos para olhar o código de nosso formulário.

```
Private Sub UserForm_Activate()  
    'Ao ativar o formulário, ativar primeira página do MultiPage  
    pageAssistente.Value = 0  
  
    'Deixar a UF em branco na página de dados pessoais  
    cboUF.ListIndex = 0  
  
    'Deixar a UF em branco na página de dados de emprego  
    cboUFEmpresa.ListIndex = 0  
  
    'Checar os controles  
    chkControles  
End Sub  
  
Private Sub cmdCancelar_Click()  
    'Descarrega o formulário  
    Unload Me  
End Sub  
  
Private Sub cmdAvançar_Click()  
    'avança uma página do MultiPage  
    pageAssistente.Value = pageAssistente.Value + 1  
  
    'Checa a condição do controle  
    chkControles  
End Sub
```

```
Private Sub cmdVoltar_Click()  
    'Volta uma página do MultiPage  
    pageAssistente.Value = pageAssistente.Value - 1  
  
    'Checa a condição do controle  
    chkControles  
End Sub  
  
Sub chkControles()  
    'guarda o índice atual do MultiPage  
    índice = pageAssistente.Value  
  
    'guarda a última página do MultiPage  
    fim = pageAssistente.Pages.Count - 1  
  
    'Verifica cada caso dos índices do MultiPage  
    Select Case índice  
        Case 0  
            'Se for a primeira página, inibe o botão "Voltar"  
            cmdVoltar.Enabled = False  
            'Habilita o botão "Avançar"  
            cmdAvançar.Enabled = True  
  
        Case fim  
            'Se for a última página, inibe o botão "Avançar"  
            cmdAvançar.Enabled = False  
            'Habilita o botão "Voltar"  
            cmdVoltar.Enabled = True  
            'Habilita o botão "Concluir"  
            cmdConcluir.Enabled = True  
  
        Case Else  
            'Para casos diferentes de início e final das  
            'páginas, habilitar os botões de "Avançar" e "Voltar"  
            cmdVoltar.Enabled = True  
            cmdAvançar.Enabled = True  
            'Desabilitar o botão de "Concluir"  
            cmdConcluir.Enabled = False  
  
    End Select  
  
    'Insere um "caption" no formulário referente a página atual do Assistente  
    frmAssistente.Caption = "Página " & índice + 1 & " de " & fim + 1  
End Sub  
  
Private Sub cboProdutoPreco_Change()  
    'Pega o salário disponível da página anterior  
    txtDisp.Value = Format(txt30pct.Value, "Currency")  
  
    'Habilita os frames da página atual  
    fraCondicoes.Enabled = True  
    fraResultados.Enabled = True  
End Sub
```

```
Private Sub opt6Meses_Click()  
    'Pega a variável referente ao prazo de pagamento.  
    'Esta variável encontra-se na propriedade "Tag" do  
    'botão de opção  
    n = Val(opt6Meses.Tag)  
  
    'Passa a variável para a Sub que calculará os valores  
    'das prestações do produto em questão  
    cálculos (n)  
End Sub
```

```
Private Sub opt12Meses_Click()  
    'Pega a variável referente ao prazo de pagamento.  
    'Esta variável encontra-se na propriedade "Tag" do  
    'botão de opção  
    n = Val(opt12Meses.Tag)  
  
    'Passa a variável para a Sub que calculará os valores  
    'das prestações do produto em questão  
    cálculos (n)  
End Sub
```

```
Private Sub opt18Meses_Click()  
    'Pega a variável referente ao prazo de pagamento.  
    'Esta variável encontra-se na propriedade "Tag" do  
    'botão de opção  
    n = Val(opt18Meses.Tag)  
  
    'Passa a variável para a Sub que calculará os valores  
    'das prestações do produto em questão  
    cálculos (n)  
End Sub
```

```
Private Sub opt24Meses_Click()  
    'Pega a variável referente ao prazo de pagamento.  
    'Esta variável encontra-se na propriedade "Tag" do  
    'botão de opção  
    n = Val(opt24Meses.Tag)  
  
    'Passa a variável para a Sub que calculará os valores  
    'das prestações do produto em questão  
    cálculos (n)  
End Sub
```

```
Private Sub opt36Meses_Click()  
    'Pega a variável referente ao prazo de pagamento.  
    'Esta variável encontra-se na propriedade "Tag" do  
    'botão de opção  
    n = Val(opt36Meses.Tag)  
  
    'Passa a variável para a Sub que calculará os valores  
    'das prestações do produto em questão  
    cálculos (n)  
End Sub
```

```
Private Sub opt48Meses_Click()  
    'Pega a variável referente ao prazo de pagamento.  
    'Esta variável encontra-se na propriedade "Tag" do  
    'botão de opção  
    n = Val(opt48Meses.Tag)  
  
    'Passa a variável para a Sub que calculará os valores  
    'das prestações do produto em questão  
    cálculos (n)  
End Sub  
  
Sub cálculos(ByVal nper)  
  
    'Dimensionamento das variáveis a serem utilizadas no cálculo  
    Dim taxa           As Double  
    Dim VP             As Double  
    Dim pgtoTotal      As Double  
    Dim ws              As Worksheet  
  
    'Define o objeto ws (worksheet) como sendo a planilha "Resultados"  
    Set ws = ThisWorkbook.Sheets("Resultados")  
  
    'Limpa o conteúdo da área que receberá os resultados do código  
    ws.Range("A2:E50").ClearContents  
  
    'Proporciona a taxa pelo valor do controle "Spin"  
    taxa = spinJuros.Value / 400  
  
    'Define o valor presente (VP) do produto  
    VP = cboProdutoPreco.Value  
  
    'Caso haja um erro, continua o processamento  
    On Error Resume Next  
  
    'Utiliza a função PMT de planilha (WorksheetFunction) para  
    'calcular o valor de cada prestação  
    pgtoTotal = WorksheetFunction.Pmt(taxa, nper, -VP, 0, 0)  
  
    'Formata o pagamento total. Você pode fazer isso no passo anterior  
    'se desejar  
    txtPgto = Format(pgtoTotal, "currency")  
  
    'Pega o quanto a pessoal tem disponível para gastar no mês  
    totalDisp = CDbl(txt30pct.Value)  
  
    'Se a parcela for maior do que o salário disponível;  
    If pgtoTotal > totalDisp Then  
        'então, o empréstimo não é autorizado.  
        lblFinanciamentoOK.Caption = "Financiamento não autorizado."  
        'Se for menor;  
    Else:  
        'então, o empréstimo é autorizado.  
        lblFinanciamentoOK.Caption = "Financiamento autorizado."  
    End If  
  
    'Coloca o valor presente na célula E2  
    ws.Range("E2") = VP  
  
    'Célula A2 = 0  
    ws.Range("A2") = 0
```

```
'Para i = 1 até nper
For i = 1 To nper
    'na planilha ws (Resultados)
    With ws
        'Célula ("A" & i + 2) = nper
        .Range("A" & i + 2) = i

        'Célula ("B" & i + 2) = ao pagamento total
        .Range("B" & i + 2) = pgtoTotal

        'Célula ("C" & i + 2) = juros pagos sobre o financiamento
        .Range("C" & i + 2) = .Range("E" & i + 1) * taxa

        'Célula ("D" & i + 2) = pgt do valor principal do financiamento
        .Range("D" & i + 2) = .Range("B" & i + 2) - .Range("C" & i + 2)

        'Célula ("E" & i + 2) = Balanço sobre o financiamento original
        .Range("E" & i + 2) = .Range("E" & i + 1) - .Range("D" & i + 2)
    End With
Next

'Passa a lista de resultados para a listbox do formulário.
'Utilizo um "nome dinâmico" para isso. Veja fórmula em:
'Inserir --> Nomes --> Definir e clique sobre "pgtos"
lstPgtos.RowSource = "pgtos"

End Sub
```

```
Private Sub spinJuros_Change()
    'Dimensiona a variável "juros"
    Dim juros As Double

    'Proporciona o juros
    juros = spinJuros.Value / 400

    'Desabilita os botões de opção e passa seus valores
    'para zero
    opt6Meses.Value = False: opt6Meses.Enabled = False
    opt12Meses.Value = False: opt12Meses.Enabled = False
    opt18Meses.Value = False: opt18Meses.Enabled = False
    opt24Meses.Value = False: opt24Meses.Enabled = False
    opt36Meses.Value = False: opt36Meses.Enabled = False
    opt48Meses.Value = False: opt48Meses.Enabled = False

    'Caso o valor da caixa de texto seja maior que zero
    'habilitar os botões de opção
    If juros > 0 Then
        opt6Meses.Enabled = True
        opt12Meses.Enabled = True
        opt18Meses.Enabled = True
        opt24Meses.Enabled = True
        opt36Meses.Enabled = True
        opt48Meses.Enabled = True
    End If

    'Formata os juros para percentuais
    txtJuros = Format(juros, "percent")

    'Calcula o pgto para nper = 0 (zero o resultado)
    cálculos (0)

End Sub
```



```

Private Sub txtDescontos_Change()
    'Verifica o tipo de dado entrado no campo "descontos"
    If Len(txtDescontos) > 0 And IsNumeric(txtDescontos) = False Then
        txtDescontos.Text = ""
        MsgBox "O valor precisa ser numérico...", vbInformation
    End If

    'Em caso de erro continuar processamento
    On Error Resume Next

    'Calcula o valor líquido
    txtLíquido = Val(txtSalBruto) - Val(txtDescontos)

    'Do valor líquido, apenas 30% pode ser comprometido com o financiamento
    txt30pct = Val(txtLíquido) * 0.3
End Sub

Private Sub txtSalBruto_Change()
    'Verifica o tipo de dado entrado no campo "Salário Bruto"
    If Len(txtSalBruto) > 0 And IsNumeric(txtSalBruto) = False Then
        txtSalBruto.Text = ""
        MsgBox "O valor precisa ser numérico...", vbInformation
    End If

    'Em caso de erro continuar processamento
    On Error Resume Next

    'Calcula o valor líquido
    txtLíquido = Val(txtSalBruto) - Val(txtDescontos)

    'Do valor líquido, apenas 30% pode ser comprometido com o financiamento
    txt30pct = Val(txtLíquido) * 0.3
End Sub

```

E nosso Assistente de Análise de Crédito está pronto para ser rodado. Na última página, você deverá obter um resultado similar a este:

Página 5 de 5

**Resultados para pagamentos**

Mês	Pqto	Juros	Principal	Balanco devido
9	R\$ 106.08	R\$ 27.15	R\$ 78.93	R\$ 825.97
10	R\$ 106.08	R\$ 24.78	R\$ 81.30	R\$ 744.66
11	R\$ 106.08	R\$ 22.34	R\$ 83.74	R\$ 660.92
12	R\$ 106.08	R\$ 19.83	R\$ 86.25	R\$ 574.67
13	R\$ 106.08	R\$ 17.24	R\$ 88.84	R\$ 485.82
14	R\$ 106.08	R\$ 14.57	R\$ 91.51	R\$ 394.32
15	R\$ 106.08	R\$ 11.83	R\$ 94.25	R\$ 300.06
16	R\$ 106.08	R\$ 9.00	R\$ 97.08	R\$ 202.98
17	R\$ 106.08	R\$ 6.09	R\$ 99.99	R\$ 102.99
18	R\$ 106.08	R\$ 3.09	R\$ 102.99	R\$ (0.00)

Cancelar < Voltar Avançar > Concluir

Figura 6-9

## 7. Carregando informações de banco de dados para os controles

Muitas vezes os usuários de Excel tendem a criar mega planilha para armazenamento e leitura de dados. Contudo, este método pode causar sérios problemas quanto a integridade dos dados.

No primeiro módulo sobre formulário, mostrei como carregar informações de um banco de dados e forma aqui apresentada é ligeiramente diferente. A intenção é ir construindo os modelos da base até o topo, isto é, iniciamos com um exemplo simples e vamos aumentando a complexidade até atingirmos um aplicativo que seja o mais robusto possível.

Uma questão que vez ou outra aparece no fórum é sobre como carregar imagens para um formulário. No exemplo que segue, mostrarei como carregar as informações de um banco de dados e como carregar uma foto também.

Os dados utilizados são do banco de dados **Northwind**. Para evitar problemas com fotos de pessoas, utilizo fotos numeradas para que você possa visualizar a mudança. Apenas uma tabela do banco de dados **Northwind** é utilizada.

As figuras abaixo mostram o que o nosso **MultiPage** deve conter e como ele ficará após o carregamento dos dados.



Figura 7-1

O formulário contém duas páginas em um **MultiPage**. A primeira página mostra informações sobre o funcionário em relação a empresa e segunda página mostra informações pessoais.

Os botões **Voltar** e **Avançar** são utilizados para navegar os registros dos funcionários. Diferentemente dos exemplos anteriores eles não são utilizados para avançar nas tabulações do **MultiPage**.

**Autor:** Robert F Martim  
**Publicado:** [www.juliobattisti.com.br](http://www.juliobattisti.com.br)  
**Contato:** [rm@faircourt.com](mailto:rm@faircourt.com)

**Criado em:** 01/02/2005  
**Última edição:** 22/3/2005 19:49:00

Para completar o exercício você precisará:

- Um **MultiPage** com duas páginas
- Um formulário
- Três botões de comando
- Sete caixas de texto
- Uma moldura de imagem
- Um módulo

O formulário será carregado assim que o documento Excel for aberto. Assim sendo, precisamos inserir o código no pasta de trabalho para a execução:

```
Private Sub Workbook_Open()  
    ' Mostra o formulário quando a planilha é aberta  
    frmBD.Show  
End Sub
```

Feito isso, o nosso próximo passo é inserir o código no formulário.

Assegure-se que os nomes utilizados são os mesmos que no exemplo dado. Se eles forem diferentes ocorrerá um erro:

```
Private Sub cmdAvançar_Click()  
    Call próximo  
End Sub
```

```
Private Sub cmdVoltar_Click()  
    Call anterior  
End Sub
```

```
Private Sub cmdFechar_Click()  
    Unload Me  
End Sub
```

```
Private Sub UserForm_Initialize()  
    ' Inicia o MultiPage sempre na primeira página  
    MultiPage1.Value = 0  
    ' Chama a sub-rotina para carregar os dados  
    Call carregarInfo  
End Sub
```

Os três primeiros casos são bem simples. O primeiro avança um registro e o segundo volta um registro. Já o terceiro simplesmente descarrega o formulário. A quarta rotina é chamada quando o formulário é iniciado assegurando que estamos sempre na primeira página do **MultiPage** e que o primeiro registro do BD é carregado.

Caso não haja dado algum o formulário é carregado em branco.

O próximo passo é definir as variáveis que usaremos. Antes de continuar, porém, instale as referências ao **Microsoft ActiveX Data Object x.x. Library**. Para tanto, no VBE, vá até **Ferramentas → Referências**, selecione o item acima e instale-o.

Com as referências instaladas, abra o módulo que você criou e entre as seguintes variáveis logo no topo:

```
Public Const Tabela      As String = "funcionários"
Public Const IDInicial   As Byte = 1

Dim cn                   As ADODB.Connection
Dim rs                   As ADODB.Recordset
Dim nomeBD               As String
Dim IDFunc               As Long
```

A primeira constante é o nome da tabela. A segunda constante é o primeiro registro da tabela. O segundo grupo de variáveis inclui uma conexão **Active Data Object (ADO)**, um **Recordset**, uma **String** para guardar o endereço e nome do banco de dados e um número longo (**long**) para armazenar o número do registro.

Se você deixou de instalar as referências ao **ADO**, você não conseguirá dimensionar as variáveis **cn** e **rs**.

Algo que precisamos observar é que a forma como os dados são carregados é igual para o início de formulário, avanço e retrocesso dos registros, portanto, não há necessidade de repetir o mesmo código várias vezes. Ao invés disso, o que faremos é passar as variáveis para o rotina que carregar os dados.

Iniciamos por estas três rotinas:

```
Sub carregarInfo()
    geral ThisWorkbook.Path & "\BD\bd2.mdb", IDInicial
End Sub

Sub próximo()
    geral ThisWorkbook.Path & "\BD\bd2.mdb", Val(frmBD.txtIDFunc) + 1
End Sub

Sub anterior()
    geral ThisWorkbook.Path & "\BD\bd2.mdb", Val(frmBD.txtIDFunc) - 1
End Sub
```

A rotina que carrega os dados para o formulário é chamada de **geral** e recebe o endereço completo do banco de dados (incluindo o nome) e o índice do funcionário. Conforme for digita o nome desta rotina ela lhe apresentará com os argumentos necessários:

```
Sub carregarInfo()
    geral
End Sub geral(nomeBD As String, IDFunc As Long)
```

Figura 7-2

Os quais, como dito, são o endereço do banco de dados (incluindo o nome) e o número serial do funcionário. É exatamente este número serial que é a chave para a navegação nos registros do banco de dados.

Com isso em mente nós podemos partir para a construção do último código. As explicações acompanham o código:

```
Sub geral(nomeBD As String, IDFunc As Long)
    'Cria a conexão
    Set cn = New ADODB.Connection

    'Abre a conexão utilizando o "Driver" correto e nome do BD
    'passado como parâmetro para esta sub-rotina
    cn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & nomeBD & ";"

    'Cria o conjunto de registros (recordset)
    Set rs = New ADODB.Recordset

    'Abre o conjunto de registros da tabela passada
    'como parâmetro para esta sub-rotina
    rs.Open "Select * FROM " & Tabela & " WHERE IDFunc=" & IDFunc, cn

    'Em caso de erro continuar execução
    On Error Resume Next
    With rs
        ' A caixa de texto para o telefone é igual ao campo "FoneRes" no BD
        frmBD.txtFone = .Fields("FoneRes")

        ' A caixa de texto para o nome é igual a concatenação dos campos
        ' "PrimeiroNome" e "Sobrenome" (incluindo um espaço entre os dois)
        frmBD.txtNome = .Fields("Título") & " " & .Fields("PrimeiroNome") _
            & " " & .Fields("Sobrenome")

        ' A mesma coisa para a caixa na segunda página do MultiPage
        frmBD.txtNome2 = .Fields("Título") & " " & .Fields("PrimeiroNome") _
            & " " & .Fields("Sobrenome")

        ' A caixa de texto para o ramal é igual ao campo "Ramal" no BD
        frmBD.txtRamal = .Fields("Ramal")

        ' A caixa de texto para o ID é igual ao campo "IDFunc" no BD
        frmBD.txtIDFunc = .Fields("IDFunc")

        ' A caixa de texto para o Cargo é igual ao campo "Cargo" no BD
        frmBD.txtCargo = .Fields("Cargo")

        ' A caixa de texto para as anotações é igual ao campo "Notas" no BD
        frmBD.txtNotas = .Fields("Notas")
    End With
End Sub
```

```
' Busca o texto referente a foto do funcionário no BD e entra como
' argumento para o carregamento da foto. O "On Error Resume Next"
' é direcionado a esta linha, pois se uma foto não for encontrada
' haverá um erro nesta linha
frmBD.fotoFunc.Picture = LoadPicture(ThisWorkbook.Path _
    & "\pics\" & .Fields("Foto"))
frmBD.fotoFunc2.Picture = LoadPicture(ThisWorkbook.Path _
    & "\pics\" & .Fields("Foto"))
End With

'Fecha o conjunto de registros
rs.Close
'Limpa o conjunto da memória
Set rs = Nothing

'Fecha a conexão ao BD
cn.Close
'Limpa a conexão da memória
Set cn = Nothing
End Sub
```

Terminado esta última rotina você está pronto para rodar o seu formulário. A nossa próxima para é como atualizar o banco de dados utilizando o mesmo modelo.

## 8. Como atualizar o banco de dados central a partir do formulário

A primeira parte mostra como carregar os dados de um banco de dados para os controles do formulário. A nossa parada agora é como atualizar informações no banco de dados.

O nosso banco de dados possui mais campos do que o nosso formulário possui de controle, contudo, deixo a adição de novos controles por conta do leitor. Isso forçará você, leitor, a criar novas possibilidades em cima deste exercício.

Aqui, utilizaremos o mesmo formulário com a mesma conexão anterior, mas faremos algumas pequenas alterações para podermos efetuar as atualizações no banco de dados.

Quando falamos em “atualizar” um banco de dados precisamos distinguir entre *inserir um registro novo* e *atualizar um registro existente*. No caso que estudaremos, abordarei o segundo caso, pois no primeiro módulo sobre formulários toquei no assunto de inserção de registro.

Para atualizar um registro existente nós utilizamos o método **Execute** de nossa conexão ao banco de dados. Para que o método funcione, precisamos montar a instrução SQL a ser executada. Para facilitar a visualização do problema quebrarei cada campo em uma linha no código e concatenarei cada instrução SQL.

Para completar o exercício, adicione mais um botão ao formulário como mostra a figura:

Figura 8-1

Ao botão **Atualizar** você deve acrescentar o seguinte código:

```
Sub atualizar()  
    geral ThisWorkbook.Path & "\BD\bd2.mdb", Val(frmBD.txtIDFunc), True  
End Sub
```

Observe que continua chamar pela mesma rotina do tópico anterior. Contudo, esta rotina agora contém mais um argumento. Este último argumento é booleano e recebe os valores **True** (verdadeiro) ou **False** (Falso). Se o valor é **True**, então, estamos atualizando o banco de dados. Se o argumento é **False**, então, estamos apenas lendo o banco de dados.

Ao passarmos este novo argumento estamos reduzindo o volume de código e rotinas necessárias.

O que precisamos fazer agora é distinguir o verdadeiro do falso para saber o que deve ser executado. Como o código é baseado no tópico anterior somente as partes novas estarão comentadas:

```

Sub geral(nomeBD As String, IDFunc As Long, atualizar As Boolean)

Set cn = New ADODB.Connection
cn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & nomeBD & ";"

Set rs = New ADODB.Recordset

On Error Resume Next
' Se (IF) não (NOT) for atualizar, então (THEN)
If Not atualizar Then
rs.Open "Select * FROM " & Tabela & " WHERE IDFunc=" & IDFunc, cn
With rs
    frmBD.txtFone = .Fields("FoneRes")
    frmBD.txtNome = .Fields("Título") & " " & .Fields("PrimeiroNome") _
        & " " & .Fields("Sobrenome")
    frmBD.txtNome2 = frmBD.txtNome
    frmBD.txtRamal = .Fields("Ramal")
    frmBD.txtIDFunc = .Fields("IDFunc")
    frmBD.txtCargo = .Fields("Cargo")
    frmBD.txtNotas = .Fields("Notas")
    frmBD.fotoFunc.Picture = LoadPicture _
        (ThisWorkbook.Path & "\pics\" & .Fields("Foto"))
    frmBD.fotoFunc2.Picture = LoadPicture _
        (ThisWorkbook.Path & "\pics\" & .Fields("Foto"))
End With
' Caso contrário (se for atualizar)
Else:

' Separa o nome nos componentes "Título", "Primeiro nome" e "Sobrenome"
' A função Split somente funciona a partir do Excel XP
nome = Split(frmBD.txtNome, " ")

' Inicia a construção da instrução SQL:
' Atualizar (UPDATE) Tabela e definir (SET) Cargo= frmBD.txtCargo ...
Sql = "UPDATE " & Tabela & " SET Cargo='" & frmBD.txtCargo & "'"
Sql = Sql & ", Título='" & nome(0) & "'"
Sql = Sql & ", PrimeiroNome='" & nome(1) & "'"
Sql = Sql & ", Sobrenome='" & nome(2) & "'"
Sql = Sql & ", Ramal='" & frmBD.txtRamal & "'"
Sql = Sql & ", Notas='" & frmBD.txtNotas & "'"
Sql = Sql & ", FoneRes='" & frmBD.txtFone & "'"
' Onde (WHERE) IDFunc= IDFunc
Sql = Sql & " WHERE IDFunc=" & IDFunc

```



```
'    Executar instrução SQL
    cn.Execute Sql

    End If

rs.Close
Set rs = Nothing

cn.Close
Set cn = Nothing

End Sub
```

Se você utilizar uma versão anterior ao Excel XP a função **Split** no código não funcionará. Observe também que a função utiliza um espaço para quebrar o texto em partes, portanto, sem o espaço a separação será errada. Além disso, no exemplo, estou assumindo apenas três partes. Se o nome for composto por primeiro nome, nome do meio e sobrenome a última parte (sobrenome) será omitida.

## 9. Removendo/adicionando informações no banco de dados e criando um sistema de navegação de registros

Este capítulo é dedicado a adição e remoção de dados no banco de dados.

Se você removeu e adicionou um item ao banco de dados anterior e tentou utilizar o sistema de navegação você certamente encontrou um erro. O código anterior assume que não haverá inserção ou remoção de registro no código e funcionará muito bem se isso for verdade.

Porém, nesta parte o código anterior não nos serve para muito. As partes comuns serão reutilizadas, mas somente isso.

Um outro problema é que se um campo encontra-se vazio no banco de dados, você notará que o valor na caixa de texto permanece o mesmo valor que o funcionário anterior na hora que navegamos pelos registros.

O primeiro passo é remontar o nosso formulário. Copie o formulário do projeto anterior (salve a pasta de trabalho com um nome diferente). Modifique o formulário como segue:



Figura 9-1

O que precisaremos fazer:

- Corrigir a navegação
- Inserir um botão para criar um registro novo
- Inserir um botão para excluir um registro

- Corrigir o problema de texto de outro registro misturado com outros registros durante a navegação

Iniciamos pelo código no formulário. As explicações são colocadas no código:

```

Private Sub cmdAtualizar_Click()
    ' chama a rotina para atualização
    Call atualizar(CLng(txtIDFunc))
End Sub

Private Sub cmdAvançar_Click()
    ' Guarda o número do registro do funcionário
    IDFunc = CLng(txtIDFunc)
    ' Chama a rotina para limpar os controles textbox
    Call limpar
    ' Chama a rotina que carrega o próximo registro
    Call próximo(IDFunc)
End Sub

Private Sub cmdVoltar_Click()
    ' Guarda o número do registro do funcionário
    IDFunc = CLng(txtIDFunc)
    ' Chama a rotina para limpar os controles textbox
    Call limpar
    ' Chama a rotina que carrega o registro anterior
    Call anterior(IDFunc)
End Sub

Private Sub cmdNovo_Click()
    ' Guarda o número do registro do funcionário
    IDFunc = CLng(txtIDFunc)
    ' Verificar o rótulo atual do botão
    ' Se o rótulo for "Novo", então
    If cmdNovo.Caption = "Novo" Then
        ' Chamar a rotina limpar
        Call limpar
        ' Muda o rótulo do botão para "Inserir"
        cmdNovo.Caption = "Inserir"
        ' Avisa o usuário o que ele deve fazer
        MsgBox "Digite os dados e clique 'Inserir' para continuar", _
            vbInformation + vbOKOnly
    Else:
        ' Se o rótulo do botão não for "Novo", então
        ' Verificar se todas as caixas de texto foram preenchidas, senão
        If Not preenchido Then
            ' avisar o usuário que ele deve preenchê-las
            MsgBox "Preencha todos os campos antes de continuar.", _
                vbInformation + vbOKOnly
            ' Sair da rotina
            Exit Sub
        End If
        ' Se todos os dados foram preenchidos, chamar a rotina "geral"
        ' A rotina "geral" recebe 5 argumentos: "nome do BD",
        ' "ID do Funcionário",
        ' "atualizar", "excluir" e "novo".
        ' Se "atualizar" for verdadeiro, então "excluir"
        ' e novo são ignorados.
        ' A mesma lógica segue para os argumentos "excluir" e "novo"
        Call geral(ThisWorkbook.Path & "\BD\bd3.mdb", IDFunc, _
            False, False, True)
        ' Passa o rótulo do botão para o valor original, isto é, "Novo"
    End If
End Sub

```

```
cmdNovo.Caption = "Novo"  
End If  
End Sub
```

```
Private Sub cmdExcluir_Click()  
    ' Verificar com o usuário se ele realmente deseja excluir o registro  
    resposta = MsgBox("Você deseja realmente excluir este registro?" _  
        & " A exclusão não pode ser desfeita!", _  
        vbQuestion + vbYesNo, "Excluir registro...")  
  
    ' Se ele apertou o botão por engano e não confirma a exclusão, então  
    ' sair da rotina  
    If Not resposta = vbYes Then Exit Sub  
  
    ' Se for confirmada a exclusão, chama a rotina excluir  
    ' A rotina excluir recebe um argumento que é o número serial  
    ' do funcionário  
    Call excluir(CLng(txtIDFunc))  
  
End Sub  
  
Private Sub cmdFechar_Click()  
    ' Fecha o formulário  
    Unload Me  
  
End Sub  
  
Private Sub MultiPage1_Change()  
    ' Define a caixa de texto 2 como sendo igual a caixa de texto 1  
    txtNome2.Text = txtNome.Text  
  
End Sub  
  
Private Sub UserForm_Initialize()  
    ' Inicia o MultiPage sempre na primeira página  
    MultiPage1.Value = 0  
    ' Chama a sub-rotina para carregar os dados  
    Call carregarInfo  
  
End Sub  
  
Private Sub limpar()  
    ' Dimensiona o controle  
    Dim ctl As Control  
    ' Procura pelos controles que contém um "tag"  
    ' igual a "txt". Veja a propriedade Tag das  
    ' caixas de texto.  
    Para cada (For Each) controle (ctl)  
        nos controles do formulário (Me.Controls)  
        For Each ctl In Me.Controls  
            ' Se o tag (If ctl.Tag) for igual a "txt", então  
            If ctl.Tag = "txt" Then  
                ' o texto do controle (ctl.Text = "")  
                ctl.Text = ""  
            ' Fecha IF  
            End If  
        ' Próximo controle  
        Next  
    End Sub
```

```

Private Function preenchido() As Boolean

    ' Dimensiona o controle
    Dim ctl As Control

    ' Assume que todos os controles estão preenchidos e,
    ' portanto, verdadeiro
    preenchido = True

    ' Para cada (For Each) controle (ctl)
    ' nos controles do formulário (Me.Controls)
    For Each ctl In Me.Controls
        ' Se o tag (If ctl.Tag) for igual a "txt", então
        If ctl.Tag = "txt" Then
            ' Se (If) o comprimento do texto do controle (Len(ctl.Texto))
            ' for zero, então
            If Len(ctl.Text) = 0 Then
                ' nem todos os controles não foram preenchidos
                ' E a função retorna falso (False)
                preenchido = False
                Exit Function
            End If
        End If
    Next
End Function

```

O botão de **atualizar** não possui um teste para saber se o usuário realmente deseja atualizar os dados. Você pode adaptar esta pequena rotina utilizando o método da rotina de exclusão de registro.

O nosso próximo passo é montar o código no módulo. No novo módulo inicie pela declaração das variáveis:

```

Public Const Tabela As String = "funcionários"
Public Const IDInicial As Byte = 1

Dim cn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim nomeBD As String
Dim IDFunc As Long

```

A seguir, nós construiremos as rotinas que são chamadas pelo formulário. Como as rotinas possuem pontos em comum, a parte a seguir deixará claro o uso dos argumentos falsos e verdadeiros colocados nas rotinas.

Iniciarei pelas rotinas gerais as quais dispensam comentários:

```

Sub carregarInfo()
    moverRegistro ThisWorkbook.Path & "\BD\bd3.mdb", IDInicial, True, False
End Sub

Sub próximo(ByVal IDFunc As Long)
    moverRegistro ThisWorkbook.Path & "\BD\bd3.mdb", IDFunc, False, False
End Sub

```

```

Sub anterior(ByVal IDFunc As Long)
    moverRegistro ThisWorkbook.Path & "\BD\bd3.mdb", IDFunc, False, True
End Sub

Sub excluir(ByVal IDFunc As Long)
    geral ThisWorkbook.Path & "\BD\bd3.mdb", IDFunc, False, True, False
End Sub

Sub atualizar(ByVal IDFunc As Long)
    geral ThisWorkbook.Path & "\BD\bd3.mdb", IDFunc, True, False, False
End Sub

```

Quando lemos, inserimos, atualizamos ou excluimos registros algumas partes são comuns a todas as rotinas acima. Desta forma, ao invés de criar uma rotina para cada uma nós criamos uma rotina geral que recebe os argumentos que necessitados. Esta rotina avaliará se o que deve ser feito é excluir, adicionar ou atualizar um registro e isto é feito pelos argumentos **True** e **False** que são passados pelas rotinas acima para a rotina geral.

```

Sub geral(nomeBD As String, ByVal IDFunc As Long, atualizar As Boolean, _
    excluir As Boolean, novo As Boolean)

    Set cn = New ADODB.Connection
    cn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & nomeBD & ";"

    Set rs = New ADODB.Recordset

    On Error Resume Next

    '   Separa o nome nos componentes "Título", "Primeiro nome" e "Sobrenome"
    '   A função Split somente funciona a partir do Excel XP
    nome = Split(frmBD.txtNome, " ")

    If atualizar = True And excluir = False And novo = False Then

        '   Inicia a construção da instrução SQL:
        '   Atualizar (UPDATE) Tabela e definir (SET) Cargo= frmBD.txtCargo ...
        Sql = "UPDATE " & Tabela & " SET Cargo='" & frmBD.txtCargo & "'"
        Sql = Sql & ", Título='" & nome(0) & "'"
        Sql = Sql & ", PrimeiroNome='" & nome(1) & "'"
        Sql = Sql & ", Sobrenome='" & nome(2) & "'"
        Sql = Sql & ", Ramal='" & frmBD.txtRamal & "'"
        Sql = Sql & ", Notas='" & frmBD.txtNotas & "'"
        Sql = Sql & ", FoneRes='" & frmBD.txtFone & "'"
        '   Onde (WHERE) IDFunc= IDFunc
        Sql = Sql & " WHERE IDFunc=" & IDFunc
        '   Executar instrução
        cn.Execute Sql

        '   Se a condição acima não for verdadeira, então avaliar esta
        '   condição. Se esta condição for verdadeira, então, remover
        '   registro atual
        ElseIf atualizar = False And excluir = True And novo = False Then
            '   String SQL usada no método Execute da conexão
            Sql = "DELETE * FROM " & Tabela & " WHERE IDFunc=" & IDFunc
            '   Executar o comando SQL
            cn.Execute Sql
            '   Recarregar as informações do formulário
            Call carregarInfo
        End If
    End If
End Sub

```

```
' Se a condição acima não for verdadeira, então avaliar esta
' condição. Se esta condição for verdadeira, então, adicionar
' registro.
ElseIf atualizar = False And excluir = False And novo = True Then
    With rs
        .Open Tabela, cn, adOpenKeyset, adLockOptimistic, adCmdTable
        .AddNew
        .Fields("Sobrenome") = nome(2)
        .Fields("PrimeiroNome") = nome(1)
        .Fields("Título") = nome(0)
        .Fields("FoneRes") = frmBD.txtFone
        .Fields("Ramal") = frmBD.txtRamal
        .Fields("Cargo") = frmBD.txtCargo
        .Fields("Notas") = frmBD.txtNotas
        .Update
    End With
    Call carregarInfo
End If

rs.Close
Set rs = Nothing

cn.Close
Set cn = Nothing

End Sub
```

Nas instruções SQL apresentadas no código acima é importante observar:

- Espaçamento. Se você observar a situação **Sql = "DELETE \* FROM " & Tabela & "WHERE IDFunc="** onde **"WHERE IDFunc="** não possui um espaçamento entre as aspas e **WHERE**, haverá um erro, pois a palavra **WHERE** estará colada no nome da **tabela**;
- Na instrução de atualização (**UPDATE**) você deve ter notado o pequeno apóstrofo. O apóstrofo é utilizado para separar o argumento a ser atualizado do campo onde a atualização ocorrerá;
- O número referente a chave primária do funcionário (**IDFunc**) não deve ser envelopada por apóstrofo como na observação anterior. Se isso for feito ocorrerá um erro e os dados não poderão ser atualizados, porque o tipo de dado passado pela instrução é diferente do contido no banco de dados

Feitas as ressalvas acima, estamos prontos para passar para a segunda parte do problema. Você deve ter observado que ao remover um registro do banco de dados o sistema de navegação pelos registros falha. Ele falha por partir do pressuposto que a numeração é serial, contínua, bastando apenas adicionar/subtrair 1 ao/do número serial atual.

Infelizmente, este nem sempre será o caso e precisamos encontrar um outra solução para o nosso problema. Quando criamos o objeto **recordset** ele nos dá várias propriedades e métodos.



Dos métodos, quatro são cruciais quando movemos dentro de um conjunto de registros (**recordset**). Estes métodos são **MoveFirst** (ir para o primeiro registro) **MoveLast** (ir para o último registro), **MoveNext** (ir para o próximo registro) e **MovePrevious** (ir para o registro anterior). O problema que encontramos com os métodos acima é que no caso do nosso formulário, uma vez que os dados foram carregados, não mais sabemos qual o registro atual além do número serial (chave primária) do usuário.

Portanto, teremos que usar este número serial para nos guiar dentro dos registros. Para fazer isso, utilizaremos um **loop** para mover os registros até que o número serial no banco de dados seja igual ao número serial no formulário. Este será o registro atual. A partir daí basta mover para frente ou para trás utilizando os métodos **MoveNext** e **MovePrevious**.

Com os devidos comentários feitos, podemos partir para o nosso código. Todas as instruções estão contidas no código:

```
Sub moverRegistro(nomeBD As String, IDFunc As Long, iniciar As Boolean, _  
    próximo As Boolean)  
  
    Set cn = New ADODB.Connection  
    cn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & nomeBD & ".mdb"  
  
    Set rs = New ADODB.Recordset  
  
    With rs  
        'Se a chama a esta rotina foi a inicial, então  
        If iniciar = True Then  
            ' Define o tipo de cursor a ser utilizado  
            .CursorType = adOpenKeyset  
            ' Abre os registro  
            .Open "Select * FROM " & Tabela, cn  
            ' Se não for o início do arquivo...  
            If Not .BOF Then  
                ' ...mover para o primeiro registro  
                .MoveFirst  
            End If  
            ' Ir para "Cont" (continuação)  
            GoTo Cont  
        End If  
  
        ' Define o tipo de cursor a ser utilizado  
        ' Por padrão o cursor é zero. Utilize cursor 1  
        ' (adOpenKeyset) para poder utilizar o método  
        ' MovePrevious. O método MovePrevious não pode ser  
        ' utilizado com o cursor 0 (adOpenForwardOnly).  
        .CursorType = adOpenKeyset  
        .Open "Select * FROM " & Tabela, cn  
  
        ' Mover para o próximo registro até que o IDFunc  
        ' no registro seja igual ao valor no formulário  
        Do Until .Fields.Item("IDFunc") = IDFunc  
            ' Quando os dois valores forem iguais,  
            ' o loop termina e a condição If abaixo é avaliada  
            .MoveNext  
        Loop  
    End With  
End Sub
```

```

' Se for para mover para o próximo registro
' (observe que o argumento "próximo" é passado como false,
' neste caso If Not próximo, passa o argumento para True e
' a linha abaixo pode ser lida como If próximo = True)
If Not próximo Then
'     Move um registro para frente a partir do registro atual
'     .MoveNext
'     Se for o último registro, então...
'     If .EOF = True Then
'         ...avisar o usuário e...
'         MsgBox "Este é o último registro! O primeiro registro " _
'             & "será mostrado", _
'             vbInformation + vbOKOnly, "Último registro..."
'         ... mover para o primeiro registro.
'         .MoveFirst
'     End If
' Se não for para mover para o próximo registro, então...
Else:
'     ...mover para o registro anterior...
'     .MovePrevious
'     Se for o primeiro do registro, então...
'     If .BOF = True Then
'         ...avisar o usuário e...
'         MsgBox "Este é o primeiro registro! O último registro " _
'             & "será mostrado", _
'             vbInformation + vbOKOnly, "Último registro..."
'         ... mover para o último registro.
'         .MoveLast
'     End If
End If

'Atualiza os valores no formulário
Cont:
On Error Resume Next
frmBD.txtFone = .Fields("FoneRes")
frmBD.txtNome = .Fields("Título") & " " & .Fields("PrimeiroNome") _
    & " " & .Fields("Sobrenome")
frmBD.txtNome2 = frmBD.txtNome
frmBD.txtRamal = .Fields("Ramal")
frmBD.txtIDFunc = .Fields("IDFunc")
frmBD.txtCargo = .Fields("Cargo")
frmBD.txtNotas = .Fields("Notas")
frmBD.fotoFunc.Picture = LoadPicture(ThisWorkbook.Path _
    & "\pics\" & .Fields("Foto"))
frmBD.fotoFunc2.Picture = LoadPicture(ThisWorkbook.Path _
    & "\pics\" & .Fields("Foto"))
End With

End Sub

```

## 10. Criando uma mini prova utilizando um sistema de navegação

O próximo exemplo mostra como criar uma mini prova utilizando um sistema de navegação com um `MultiPage`.

Existem várias formas de se levar a cabo tal proposição, porém, aqui farei esta combinação para demonstrar como o sistema de navegação pode ser utilizado para diversas finalidades.

Esta parte será bem curta, pois tudo que necessitamos já foi desenvolvido anteriormente. Estarei apenas mostrando como se colocar em prática o que foi ensinado.

Para iniciar, podemos colocar as perguntas e respostas na planilha ou digitá-las diretamente no formulário. Digitaremos as perguntas diretamente no formulário e as respostas na planilha. Primeiramente, iremos criar esta planilha contendo as informações que precisaremos. O formato geral será:

	C6		$\text{fx} = \text{C5}/4$			
	A	B	C	D	E	F
1	C				0%	Você precisa estudar muito ainda!
2	B				25%	Você tem certeza que estava preparado para a prova?
3	D				50%	O seu aproveitamento foi razoável.
4	B				75%	O seu aproveitamento foi acima da média! Parabéns!
5					100%	Você é um gênio!!
6		Acertos	0%			
7						

Figura 10-1

No intervalo `A1:A4` encontra-se a letra da resposta correta em nosso formulário. No intervalo `C1:C4` colocaremos as respostas do usuário. O percentual de acerto é a proporção dos acertos em relação ao número de perguntas e é calculado na célula `C6`. O intervalo `E1:E5` refere-se ao percentuais de acerto e `F1:F5` contém o texto que será mostrado após a marcação da prova.

Com esta parte pronto, precisamos apenas construir o nosso formulário que servirá para prova. O formulário possui um `MultiPage` com quatro páginas e deve ter um visual similar a este:

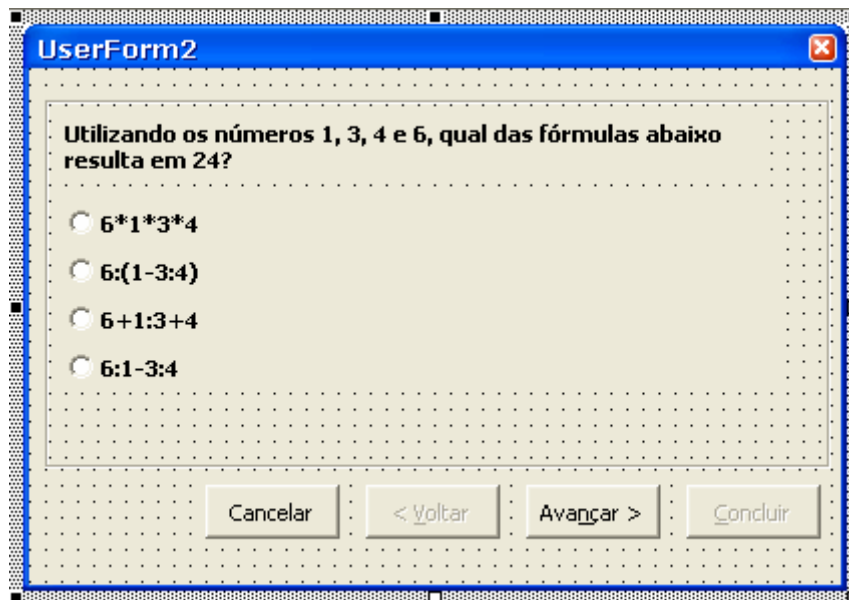


Figura 10-2

Para os botões Cancela, Voltar, Avançar e Concluir, nós utilizaremos os seguintes códigos<sup>3</sup>:

```
Private Sub cmdCancelar_Click()  
    resp = MsgBox("Você tem certeza que deseja cancelar?", _  
        vbYesNo + vbQuestion)  
    If resp = vbYes Then  
        Unload Me  
    End If  
End Sub  
  
Private Sub cmdAvançar_Click()  
    pageAssistente.Value = pageAssistente.Value + 1  
    chkControles  
End Sub  
  
Private Sub cmdVoltar_Click()  
    MultiPage1.Value = MultiPage1.Value - 1  
    chkPaginação  
End Sub
```

<sup>3</sup> Incluso no esquema também estão o código de verificação de paginação.

```
Sub chkControles()  
  
    índice = pageAssistente.Value  
    Fim = pageAssistente.Pages.Count - 1  
  
    Select Case índice  
        Case 0  
            cmdVoltar.Enabled = False  
            cmdAvançar.Enabled = True  
  
        Case Fim  
            cmdVoltar.Enabled = True  
            cmdAvançar.Enabled = False  
            cmdConcluir.Enabled = True  
  
        Case Else  
            cmdVoltar.Enabled = True  
            cmdAvançar.Enabled = True  
            cmdConcluir.Enabled = False  
  
    End Select  
  
    frmAssistente.Caption = "Página " & índice + 1 & " de " & Fim + 1  
  
End Sub
```

Como sistema de navegação pronto, podemos partir para a solução dos próximos problemas. Primeiramente, precisamos nos atentar para a planilha onde colocaremos as respostas que o usuário der.

A propriedade `visible` será passada para `xlSheetVeryHidden` para evitar acesso pelo usuário. Porém, precisamos lembrar que os dados são inseridos nesta planilha, portanto, precisamos de uma variável que representa a nossa planilha. Fora de qualquer procedimento dentro do formulário entraremos a seguinte linha de código (no topo da janela de código):

```
Dim ws As Worksheet
```

A variável `ws` refere-se a uma planilha qualquer. O nosso próximo trabalho é instanciar esta variável como sendo a planilha que contém os dados que utilizaremos e na qual escreveremos as respostas do usuário:

```
Private Sub UserForm_Activate()  
    ' "Seta" a planilha correspondente às respostas  
    Set ws = ThisWorkbook.Sheets("Respostas")  
  
    'Vai para a primeira página do assistente  
    pageAssistente.Value = 0  
  
    'checa os controles  
    chkControles  
  
    'Limpa a área contendo as respostas  
    ws.Range("C1:C4").Clear  
  
End Sub
```

Cada botão de opção tem um valor próprio que é passado para a planilha conforme o clique de quem está fazendo a prova:

'Passa os valores de cada botão de opção para  
'a planilha dos resultados

```
Private Sub optQ1A_Click()  
    ws.Range("C1") = 0  
End Sub
```

```
Private Sub optQ1B_Click()  
    ws.Range("C1") = 0  
End Sub
```

```
Private Sub optQ1C_Click()  
    ws.Range("C1") = 1  
End Sub
```

```
Private Sub optQ1D_Click()  
    ws.Range("C1") = 0  
End Sub
```

```
Private Sub optQ2A_Click()  
    ws.Range("C2") = 0  
End Sub
```

```
Private Sub optQ2B_Click()  
    ws.Range("C2") = 1  
End Sub
```

```
Private Sub optQ2C_Click()  
    ws.Range("C2") = 0  
End Sub
```

```
Private Sub optQ2D_Click()  
    ws.Range("C2") = 0  
End Sub
```

```
Private Sub optQ3A_Click()  
    ws.Range("C3") = 0  
End Sub
```

```
Private Sub optQ3B_Click()  
    ws.Range("C3") = 0  
End Sub
```

```
Private Sub optQ3C_Click()  
    ws.Range("C3") = 0  
End Sub
```

```
Private Sub optQ3D_Click()  
    ws.Range("C3") = 1  
End Sub
```

```
Private Sub optQ4A_Click()  
    ws.Range("C4") = 0  
End Sub
```

```

Private Sub optQ4B_Click()
    ws.Range("C4") = 1
End Sub

```

```

Private Sub optQ4C_Click()
    ws.Range("C4") = 0
End Sub

```

```

Private Sub optQ4D_Click()
    ws.Range("C4") = 0
End Sub

```

O último passo é o código para o botão de conclusão da prova:

```

Private Sub cmdConcluir_Click()

    j = 0
    'Faz um loop pelas células que contém as respostas
    For i = 1 To 4
        'Se não houver resposta
        If ws.Cells(i, 3) = "" Then
            'então, compile as perguntas não respondidas
            txt = txt & i & ", "
            j = j + 1
        End If
    Next

    'Se a variável j > 0
    If j > 0 Then
        'então, há perguntas não respondidas
        resposta = MsgBox("Você não respondeu a(s) pergunta(s): " & _
            txt & "você realmente deseja marcar a prova?", _
            vbYesNo + vbQuestion)
        'Se o aluno não deseja terminar, sair da Sub-rotina
        'e continuar prova
        If resposta = vbNo Then Exit Sub
    End If

    'Resultado em percentuais
    res = Format(ws.Range("C6"), "Percent")

    'Procura o tipo de parabenização para mostrar ao aluno.
    txt = WorksheetFunction.VLookup(ws.Range("C6"), ws.Range("E1:F5"), 2, 0)

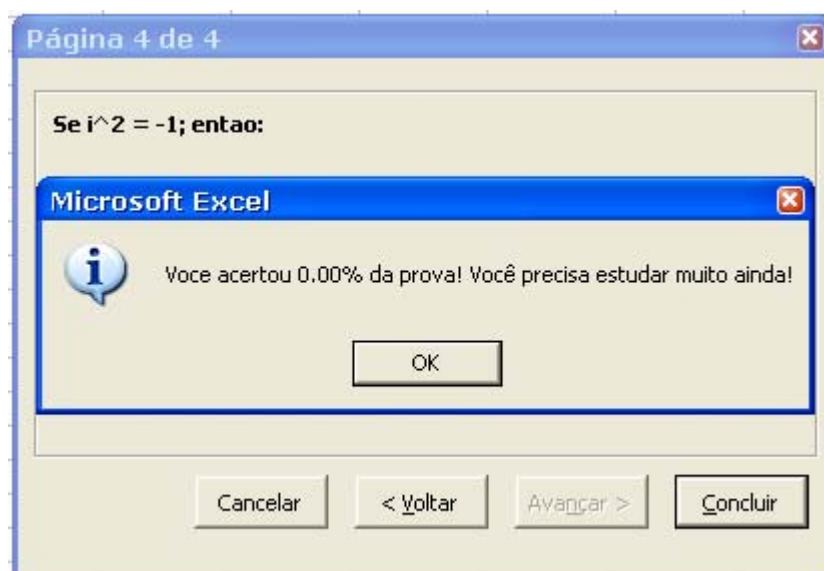
    'Mostra ao aluno uma mensagem com o percentual de acerto
    MsgBox "Voce acertou " & res & " da prova! " & txt, vbInformation

    'Descarrega o formulário
    Unload Me

End Sub

```

Ao concluir a prova o aluno recebe sua nota final através de uma caixa de mensagens:



**Figura 10-3 – Resultado final da prova**

No meu caso, não fui “muito bem” na prova, tendo um aproveitamento igual a zero!



## 11. Criando um formulário em diferentes idiomas em tempo de execução

Se você alguma vez desenvolveu algo para usuários que falam diferentes idiomas, você deve ter esbarrado no problema da criação de um projeto multi-idíomas.

Obviamente que o pré-requisito da criação de um projeto em vários idiomas é o conhecimento dos idiomas em questão. Sem isso, não temos como continuar! Ou será que temos? Hoje, com as facilidades proporcionadas pela tecnologia, nós podemos utilizar tradutores eletrônicos para resolver nosso problema.

Como estamos interessados em saber apenas palavras-chave, tradutores eletrônicos resolvem bem o nosso caso. No exemplo que darei, será criado um formulário em três idiomas: Português, Inglês e Alemão. Para referência do leitor, as traduções para o Alemão foram feitas no site [http://www.worldlingo.com/en/products\\_services/worldlingo\\_translator.html](http://www.worldlingo.com/en/products_services/worldlingo_translator.html).

O que será necessário para este exercício:

- Criar um banco de dados no Access contendo uma tabela e os seguintes campos:
  - Nome do controle
  - Português
  - Inglês
  - Alemão
- Um formulário no Excel

O campo para o nome do controle conterá o nome de cada controle que contém um “caption” em nosso formulário. Os nomes entrados neste campo devem ser exatamente iguais aos nomes dos controles no formulário.

Ao lado de cada nome conterá a tradução do “caption” do controle como mostra a figura abaixo:

	códItem	NomeCtl	Portugues	Ingles	Alemao
	1	lblNome	Nome do funcionário	Employee Name	Angestellt-Name
	2	lblTel	Telefone residência	Home Tel. Number	Persönliche Zahl
	3	lblEndereco	Endereço	Address	Adresse
	4	lblCidade	Cidade	City	Stadt
	5	lblAnot	Anotações	Notes	Anmerkungen
	6	cmdNovo	Novo Registro	New Record	Neue Aufzeichnung
	7	cmdApagar	Del. Registro	Del. Record	Löschungaufzeichnung
	8	cmdFechar	Fechar	Close	Schließen
	9	lblCEP	CEP	ZIP Code	Postcode
	10	cmdProx	Próximo >>	Next >>	Zunächst >>
►	(AutoNumber)				

Figura 11-1

No banco de dados que acompanha este módulo há também o campo `códItem`, como mostra a figura acima. Este campo é desnecessário ao nosso exercício. Eu apenas o acrescento por força de hábito, pois chaves-primárias são vitais em bancos de dados.

A construção do layout geral do formulário fica a critério do leitor. O formulário que utilizarei é feito sobre um MultiPage. A primeira página do MultiPage contém os idiomas disponíveis:



Figura 11-2

As tabulações do MultiPage são removidas para deixar o visual geral mais belo.

Observe que a grande vantagem deste sistema é que os recursos são externos. Se você precisa inserir mais controles e necessita de mais traduções, não há porque esquentar a cabeça. Os dados inseridos no banco de dados independem de ordem. Tudo que precisamos é entrar o nome

do controle no banco de dados e a respectiva tradução. Não precisamos nem modificar o “caption” padrão do controle (ou até mesmo o nome, se você desejar).

O idioma padrão é o português e todos os rótulos (labels) já estão escritos em português. A segunda página do MultiPage tem a seguinte aparência e é acessada pelo botão “Próximo”:

A screenshot of a VBA form titled "Formulário multilingue...". The form has a light beige background and a blue border. It contains several input fields: "Nome do funcionário" (a text box with a dropdown arrow), "Telefone residência" (a text box), "Endereço" (a large text box), "Cidade" (a text box), and "CEP" (a text box). There is also a large text box labeled "Anotações". At the bottom, there are three buttons: "Novo Registro", "Del. Registro", and "Fechar".

Figura 11-3

Com o layout geral pronto, podemos iniciar a escrita do código. Antes de iniciar, porém, é necessário instalar Microsoft ActiveX Data Object xx Library. Para instalá-lo, no VBE, vá até Ferramentas → Referências e da lista que aparecer, procure pela referência acima e a instale.

#### Código na Pasta de Trabalho

```
Private Sub Workbook_Open()  
    'Mostra o formulário quando a planilha é aberta  
    frmDB.Show  
End Sub
```

**Código no Formulário**

```
Private Sub UserForm_Activate()  
    'Seleciona a página inicial do MultiPage  
    MultiPage1.Value = 0  
End Sub  
  
Private Sub cmdProx_Click()  
    'Move para a segunda página do MultiPage  
    MultiPage1.Value = MultiPage1.Value + 1  
End Sub  
  
Private Sub imgBrasil_Click()  
    'Chama a sub-rotina para carregar as traduções  
    'A sub-rotina recebe 3 argumentos:path e nome BD, _  
    'nome da tabela e o idioma  
    traduzir ThisWorkbook.Path & "\BD\bd1.mdb", "ctlidiomas", _  
    "Portugues"  
End Sub  
  
Private Sub imgAle_Click()  
    'Chama a sub-rotina para carregar as traduções  
    'A sub-rotina recebe 3 argumentos:path e nome BD, _  
    'nome da tabela e o idioma  
    traduzir ThisWorkbook.Path & "\BD\bd1.mdb", "ctlidiomas", "Alemao"  
End Sub  
  
Private Sub imgEUA_Click()  
    'Chama a sub-rotina para carregar as traduções  
    'A sub-rotina recebe 3 argumentos:path e nome BD, _  
    'nome da tabela e o idioma  
    traduzir ThisWorkbook.Path & "\BD\bd1.mdb", "ctlidiomas", "Ingles"  
End Sub
```

Código no Módulo

```

Sub traduzir(nomeBD As String, Tabela As String, Idioma As String)

'Define a variável (objeto) da conexão ao banco de dados
Dim conn As ADODB.Connection

'Define a variável (objeto) do conjunto de registros
Dim rs As ADODB.Recordset

'Define o controle (objeto) a ser traduzido
Dim ctl As Control

'Cria a conexão
Set conn = New ADODB.Connection

'Abre a conexão utilizando o "Driver" correto e nome do BD
'passado como parâmetro para esta sub-rotina
conn.Open "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=" & nomeBD & ";"

'Cria o conjunto de registros (recordset)
Set rs = New ADODB.Recordset

'Abre o conjunto de registros da tabela passada
'como parâmetro para esta sub-rotina
rs.Open "Select * FROM " & Tabela, conn

'Faz o loop para cada controle no formulário
For Each ctl In frmDB.Controls

'Move para o primeiro registro do BD
rs.MoveFirst
'Em caso de erro, a sub-rotina continua
'a execução do código
On Error Resume Next
'Verifica se o "caption" não está vazio
If ctl.Caption <> "" Then
    'Se não estiver, então inicia o processo
    'de tradução
    With rs
        'Faz um loop enquanto nao chegar ao
        'fim do recordset (EOF=End of File)
        'End of File = Fim do Arquivo
        While Not .EOF
            'Compara o nome do controle com o
            'nome no banco de dados
            If ctl.Name = .Fields("NomeCtl") Then
                'Se nomes são iguais; então, traduz
                ctl.Caption = .Fields(Idioma)
                'E sai do loop para evitar perda
                'de tempo com excesso de loop
                GoTo cont
            End If
            'Move para o próximo registro
            .MoveNext
        Wend
    End With
    End If
    cont:
Next

```

```
'Fecha o conjunto de registros  
rs.Close  
'Limpa o conjunto da memória  
Set rs = Nothing  
  
'Fecha a conexão ao BD  
conn.Close  
'Limpa a conexão da memória  
Set conn = Nothing  
End Sub
```

Agora, basta rodar o nosso pequeno aplicativo e selecionar o idioma desejado:

The screenshot shows a VBA form titled "Formulário multilingue...". The form has a light beige background and a blue border. It contains the following fields and controls:

- Nome do funcionário**: A text box with a dropdown arrow on the right.
- Telefone residência**: A text box.
- Endereço**: A large text box.
- Cidade**: A text box.
- CEP**: A text box.
- Anotações**: A large text box.
- Buttons**: Three buttons at the bottom: "Novo Registro", "Del. Registro", and "Fechar".

Figura 11-4 – Português

The screenshot shows a VBA UserForm titled "Formulário multilingue...". It contains the following fields and controls:

- Angestellt-Name**: A text box with a dropdown arrow.
- Persönliche Zahl**: A text box.
- Adresse**: A large text box.
- Stadt**: A text box.
- Postcode**: A text box.
- Anmerkungen**: A large text box.
- Buttons**: Three buttons at the bottom labeled "Neue Aufzeichnung", "Löschungaufzeichnung", and "Schließen".

Figura 11-5 – Alemão

The screenshot shows the same VBA UserForm titled "Formulário multilingue..." but with English labels:

- Employee Name**: A text box with a dropdown arrow.
- Home Tel. Number**: A text box.
- Address**: A large text box.
- City**: A text box.
- ZIP Code**: A text box.
- Notes**: A large text box.
- Buttons**: Three buttons at the bottom labeled "New Record", "Del. Record", and "Close".

Figura 11-6 – Inglês

## 12. Sobre o autor

Abaixo o leitor encontra um pequeno resumo do currículo e atividades do autor deste módulo:

### **FORMAÇÃO ACADÊMICA:**

- Formado e Pós-Graduado em Finanças pela Universidade de Londres, Reino Unido
- Membro da Sociedade Brasileira de Econometria

### **LINGUAGENS DE PROGRAMAÇÃO E PLATAFORMAS:**

- Visual Basic, Calculadores Programáveis Casio e Sharp
- BDs: MS Access and Lotus Approach
- Plataformas: Windows NT, 2000, XP, Linux Red Hat

### **EXPERIÊNCIA PROFISSIONAL**

outubro 02- **FAIRCOURT CAPITAL LIMITED (REINO UNIDO)**

- ***Diretor TI***

fev96-maio02 **MELVALE GROUP (REINO UNIDO)**

- ***Gerente de Exportação para a África Ocidental***
- ***Gerente de TI***

### **OUTRAS ESPECIALIZAÇÕES**

- Inspeção e regulamentações Nigerianas para importação e exportação (Nigerian-British Chamber of Commerce & Cotecna International)
- Procedimentos de exportação no Reino Unido (The Institute of Export, Reino Unido)
- ICC 500 e Incoterms (The Institute of Export, Reino Unido)

### **OUTRAS ATIVIDADES**

Fornece suporte *pro bono* em TI à entidade de caridade Nigeriana NIDOE (Nigerians in Diaspora Organisation Europe) desde 2001. Participou ativamente na organização da conferência sobre Boa Governança e Responsabilidade Fiscal promovida pelo ONG em Abuja, Nigéria, em Novembro 2003. Foi um dos principais colaboradores na elaboração do relatório final sobre a conferência entregue a presidência da República Nigeriana em maio de 2004.

Autor do livro Access e VBA na Modelagem Financeira: Uma abordagem prática (no prelo). Editora Axcel Books, 2005.

Colaborador ativo do fórum Access Avançado do site [www.juliobattisti.com.br](http://www.juliobattisti.com.br), onde divide seu conhecimento e experiência com outros membros do espaço.

Colunista dos sites [www.linhadecodigo.com.br](http://www.linhadecodigo.com.br) e [www.ativoaccess.com.br](http://www.ativoaccess.com.br)

**Autor:** Robert F Martim  
**Publicado:** [www.juliobattisti.com.br](http://www.juliobattisti.com.br)  
**Contato:** [rm@faircourt.com](mailto:rm@faircourt.com)

**Criado em:** 01/02/2005  
**Última edição:** 22/3/2005 19:49:00